**Mandatory HandsOn**

**Spring Core Maven**

**Exercise 1: Configuring a Basic Spring Application**

**Scenario:**

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**Steps:**

1. Set Up a Spring Project:
   * Create a Maven project named LibraryManagement.
   * Add Spring Core dependencies in the pom.xml file.
2. Configure the Application Context:
   * Create an XML configuration file named applicationContext.xml in the src/main/resources directory.
   * Define beans for BookService and BookRepository in the XML file.
3. Define Service and Repository Classes:
   * Create a package com.library.service and add a class BookService.
   * Create a package com.library.repository and add a class BookRepository.
4. Run the Application:
   * Create a main class to load the Spring context and test the configuration.

**Ans. BookService.java**

package com.example.service;

import com.example.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook() {

        System.out.println("Adding a book");

        bookRepository.save();

    }

}

**BookRepository.java**

package com.example.repository;

public class BookRepository {

    public void save() {

        System.out.println("Book saved to the database.");

    }

}

**App.java**

package com.example;

import com.example.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = context.getBean("bookService", BookService.class);

        bookService.addBook();

    }

}

**applicationcontext.xml**

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

       xsi:schemaLocation="http://www.springframework.org/schema/beans

           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="bookRepository" class="com.example.repository.BookRepository" />

    <bean id="bookService" class="com.example.service.BookService">

        <property name="bookRepository" ref="bookRepository"/>

    </bean>

</beans>

**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.example</groupId>

    <artifactId>demo</artifactId>

    <version>1.0</version>

    <properties>

        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

        <maven.compiler.source>1.8</maven.compiler.source>

        <maven.compiler.target>1.8</maven.compiler.target>

    </properties>

    <dependencies>

        <!-- Spring Context -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

            <version>5.3.29</version>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Exec Plugin -->

            <plugin>

                <groupId>org.codehaus.mojo</groupId>

                <artifactId>exec-maven-plugin</artifactId>

                <version>3.1.0</version>

                <configuration>

                    <mainClass>com.example.App</mainClass>

                </configuration>

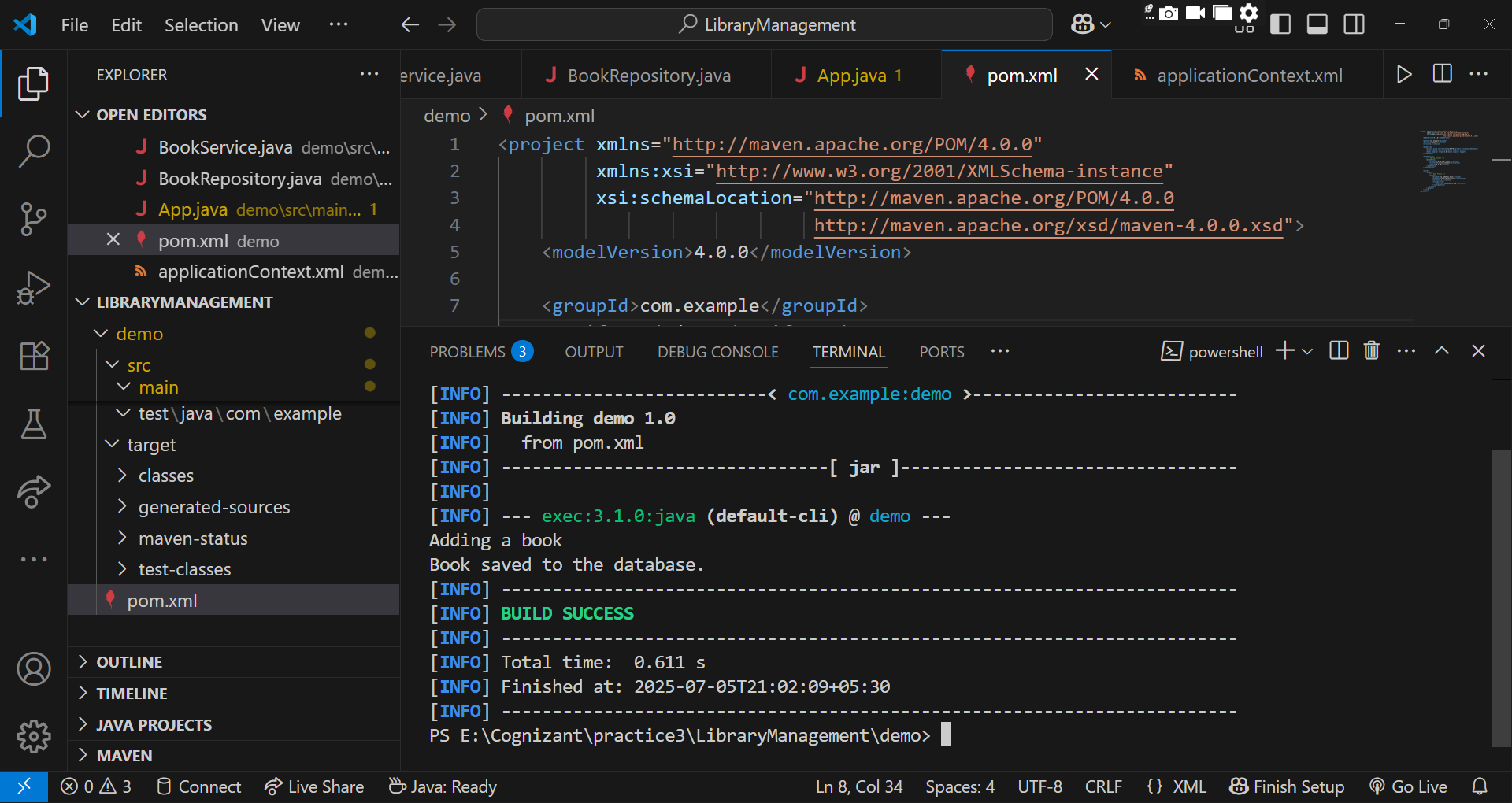
            </plugin>

        </plugins>

    </build>

</project>

**OUTPUT:**

****

**Exercise 2: Implementing Dependency Injection**

**Scenario:**

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

**Steps:**

1. Modify the XML Configuration:
   * Update applicationContext.xml to wire BookRepository into BookService.
2. Update the BookService Class:
   * Ensure that BookService class has a setter method for BookRepository.
3. Test the Configuration:
   * Run the LibraryManagementApplication main class to verify the dependency injection.

Ans.

**Book.java**

package com.example.library;

public class Book {

    private String title;

    public Book(String title) {

        this.title = title;

    }

    public String getTitle() {

        return title;

    }

}

**BookRepository.java**

package com.example.library;

public class BookRepository {

    public void save(Book book) {

        System.out.println("Saving book: " + book.getTitle());

    }

}

**BookService.java**

package com.example.library;

public class BookService {

    private BookRepository bookRepository;

    // Setter method for DI

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook(String title) {

        Book book = new Book(title);

        bookRepository.save(book);

    }

}

**LibraryMangementApplication.java**

package com.example.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = (BookService) context.getBean("bookService");

        bookService.addBook("Harry Potter and the Sorcerer's Stone");

    }

}

**applicationContext.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

       xsi:schemaLocation="

           http://www.springframework.org/schema/beans

           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Define BookRepository bean -->

    <bean id="bookRepository" class="com.example.library.BookRepository" />

    <!-- Define BookService bean and inject BookRepository -->

    <bean id="bookService" class="com.example.library.BookService">

        <property name="bookRepository" ref="bookRepository" />

    </bean>

</beans>

**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

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

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>

  <artifactId>library-management</artifactId>

  <version>1.0-SNAPSHOT</version>

  <name>LibraryManagement</name>

  <properties>

    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <maven.compiler.source>1.8</maven.compiler.source>

    <maven.compiler.target>1.8</maven.compiler.target>

  </properties>

  <dependencies>

    <!-- Spring Core -->

    <dependency>

      <groupId>org.springframework</groupId>

      <artifactId>spring-core</artifactId>

      <version>5.3.30</version>

    </dependency>

    <!-- Spring Context -->

    <dependency>

      <groupId>org.springframework</groupId>

      <artifactId>spring-context</artifactId>

      <version>5.3.30</version>

    </dependency>

    <!-- JUnit for testing (optional) -->

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>4.13.2</version>

      <scope>test</scope>

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <!-- Compiler Plugin -->

      <plugin>

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

        <version>3.8.0</version>

        <configuration>

          <source>1.8</source>

          <target>1.8</target>

        </configuration>

      </plugin>

      <!-- Exec Plugin to run your main class -->

      <plugin>

        <groupId>org.codehaus.mojo</groupId>

        <artifactId>exec-maven-plugin</artifactId>

        <version>3.5.1</version>

        <configuration>

          <mainClass>com.example.library.LibraryManagementApplication</mainClass>

        </configuration>

      </plugin>

    </plugins>

  </build>

</project>

**OUTPUT:**

****

**Exercise 4: Creating and Configuring a Maven Project**

**Scenario:**

You need to set up a new Maven project for the library management application and add Spring dependencies.

**Steps:**

1. Create a New Maven Project:
   * Create a new Maven project named LibraryManagement.
2. Add Spring Dependencies in pom.xml:
   * Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. Configure Maven Plugins:
   * Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

**Ans.**

**LibraryManagmentApplication.java**

package com.cognizant.library;

import com.cognizant.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = context.getBean(BookService.class);

        bookService.showBook();

    }

}

**Book.java**

package com.cognizant.library.model;

public class Book {

    private String title;

    private String author;

    public Book() {}

    public Book(String title, String author) {

        this.title = title;

        this.author = author;

    }

    public void display() {

        System.out.println("Book: " + title + " by " + author);

    }

}

**BookRepository.java**

package com.cognizant.library.repository;

import com.cognizant.library.model.Book;

public class BookRepository {

    public Book getBook() {

        return new Book("Clean Code", "Robert C. Martin");

    }

}

**BookService.java**

package com.cognizant.library.service;

import com.cognizant.library.model.Book;

import com.cognizant.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter Injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void showBook() {

        Book book = bookRepository.getBook();

        book.display();

    }

}

**applicationContext.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

       xsi:schemaLocation="http://www.springframework.org/schema/beans

                           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Repository Bean -->

    <bean id="bookRepository" class="com.cognizant.library.repository.BookRepository"/>

    <!-- Service Bean with Setter Injection -->

    <bean id="bookService" class="com.cognizant.library.service.BookService">

        <property name="bookRepository" ref="bookRepository"/>

    </bean>

</beans>

**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.library</groupId>

    <artifactId>LibraryManagement</artifactId>

    <version>1.0</version>

    <properties>

        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

        <maven.compiler.source>1.8</maven.compiler.source>

        <maven.compiler.target>1.8</maven.compiler.target>

    </properties>

    <dependencies>

        <!-- Spring Core & Context -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

            <version>5.3.29</version>

        </dependency>

        <!-- Spring AOP -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-aop</artifactId>

            <version>5.3.29</version>

        </dependency>

        <!-- Spring WebMVC -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-webmvc</artifactId>

            <version>5.3.29</version>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Compiler Plugin -->

            <plugin>

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

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

                <version>3.8.1</version>

                <configuration>

                    <source>1.8</source>

                    <target>1.8</target>

                </configuration>

            </plugin>

            <!-- Exec Plugin for running the main class -->

            <plugin>

                <groupId>org.codehaus.mojo</groupId>

                <artifactId>exec-maven-plugin</artifactId>

                <version>3.1.0</version>

                <configuration>

                    <mainClass>com.cognizant.library.LibraryManagementApplication</mainClass>

                </configuration>

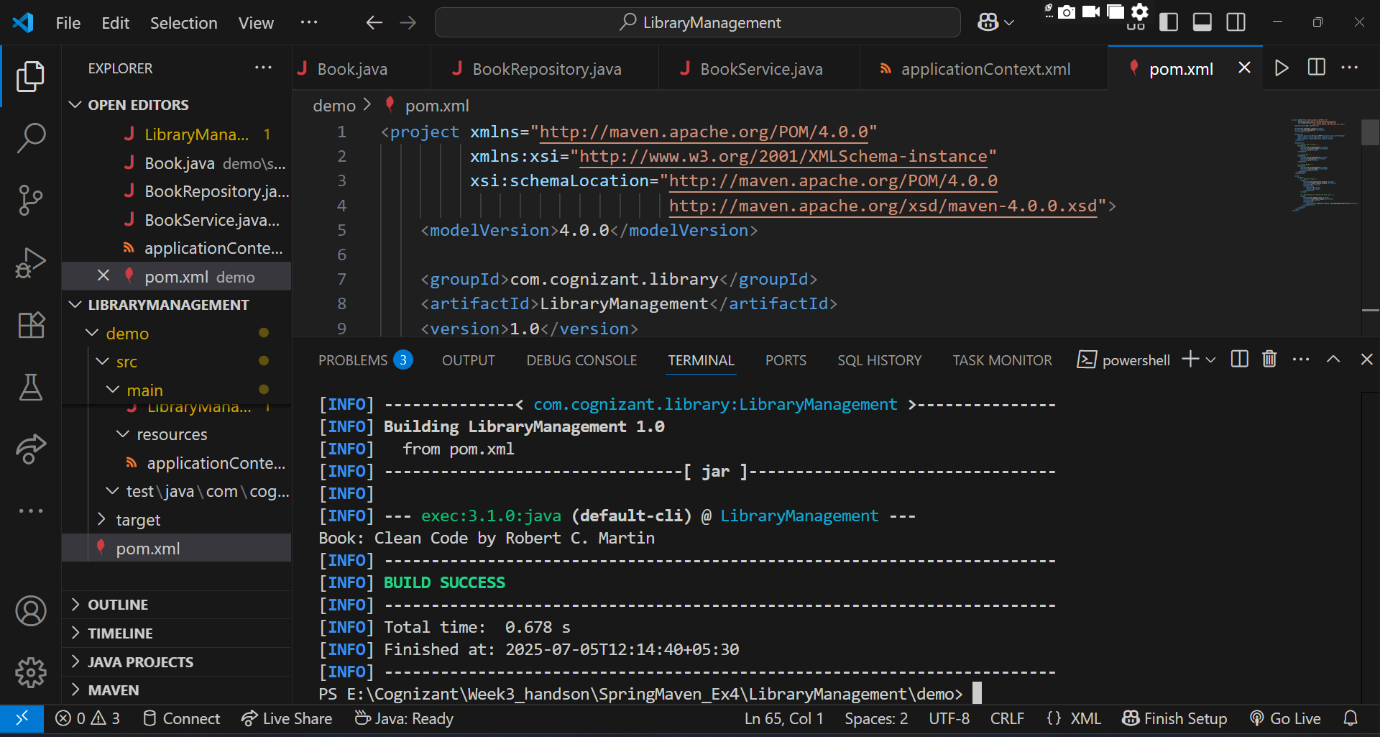
            </plugin>

        </plugins>

    </build>

</project>

**Output:**

****

**Spring Data JPA HandsOn**

**Hands on 1**

**Spring Data JPA - Quick Example**

**OrmLearnApplication.java**

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

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);

        LOGGER.info("Inside main");

        countryService = context.getBean(CountryService.class);

        testGetAllCountries();

    }

    private static void testGetAllCountries() {

        LOGGER.info("Start");

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

        if (countries.isEmpty()) {

            LOGGER.warn("No countries found in the database.");

        } else {

            for (Country country : countries) {

                LOGGER.debug("Country: {}", country);

            }

        }

        LOGGER.info("End");

    }

}

**Country.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

    @Id

    @Column(name = "code")

    private String code;

    @Column(name = "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 + "]";

    }

}

**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> {

}

**CountryService.java**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service

public class CountryService {

    @Autowired

    private CountryRepository countryRepository;

    @Transactional

    public List<Country> getAllCountries() {

        return countryRepository.findAll();

    }

}

**application.properties**

server.port=9090

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=create

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

logging.level.org.hibernate.SQL=debug

logging.level.org.hibernate.type.descriptor.sql=trace

logging.level.org.springframework=info

logging.level.com.cognizant=debug

**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>springlearn-country-lookup</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>springlearn-country-lookup</name>

    <description>Spring Boot Country Lookup with H2</description>

    <parent>

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

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

        <version>3.2.4</version>

        <relativePath/>

    </parent>

    <properties>

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

    </properties>

    <dependencies>

        <dependency>

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

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

        </dependency>

        <dependency>

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

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

        </dependency>

        <dependency>

            <groupId>com.h2database</groupId>

            <artifactId>h2</artifactId>

            <scope>runtime</scope>

        </dependency>

        <dependency>

            <groupId>jakarta.persistence</groupId>

            <artifactId>jakarta.persistence-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.springframework.boot</groupId>

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

            </plugin>

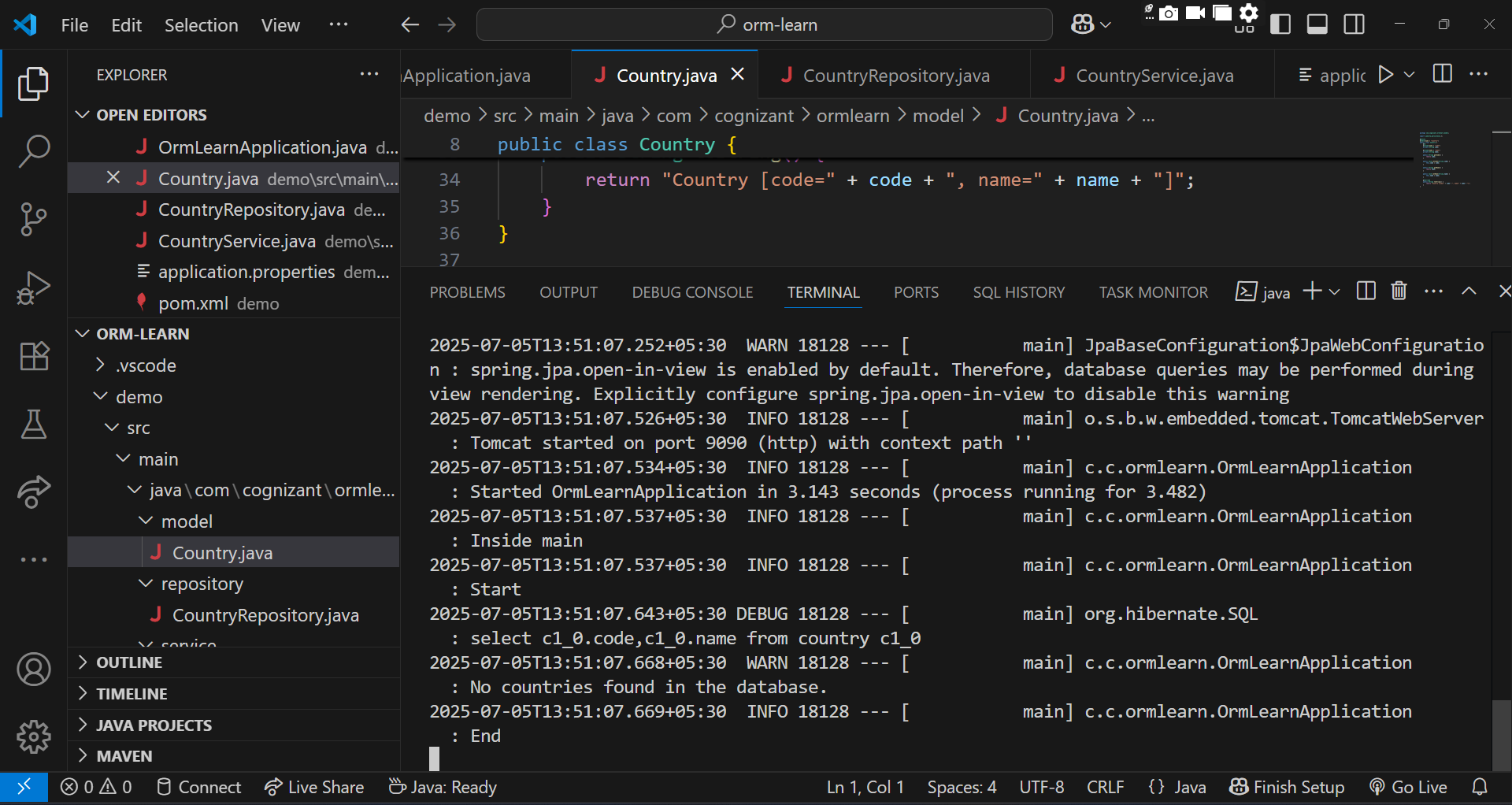
        </plugins>

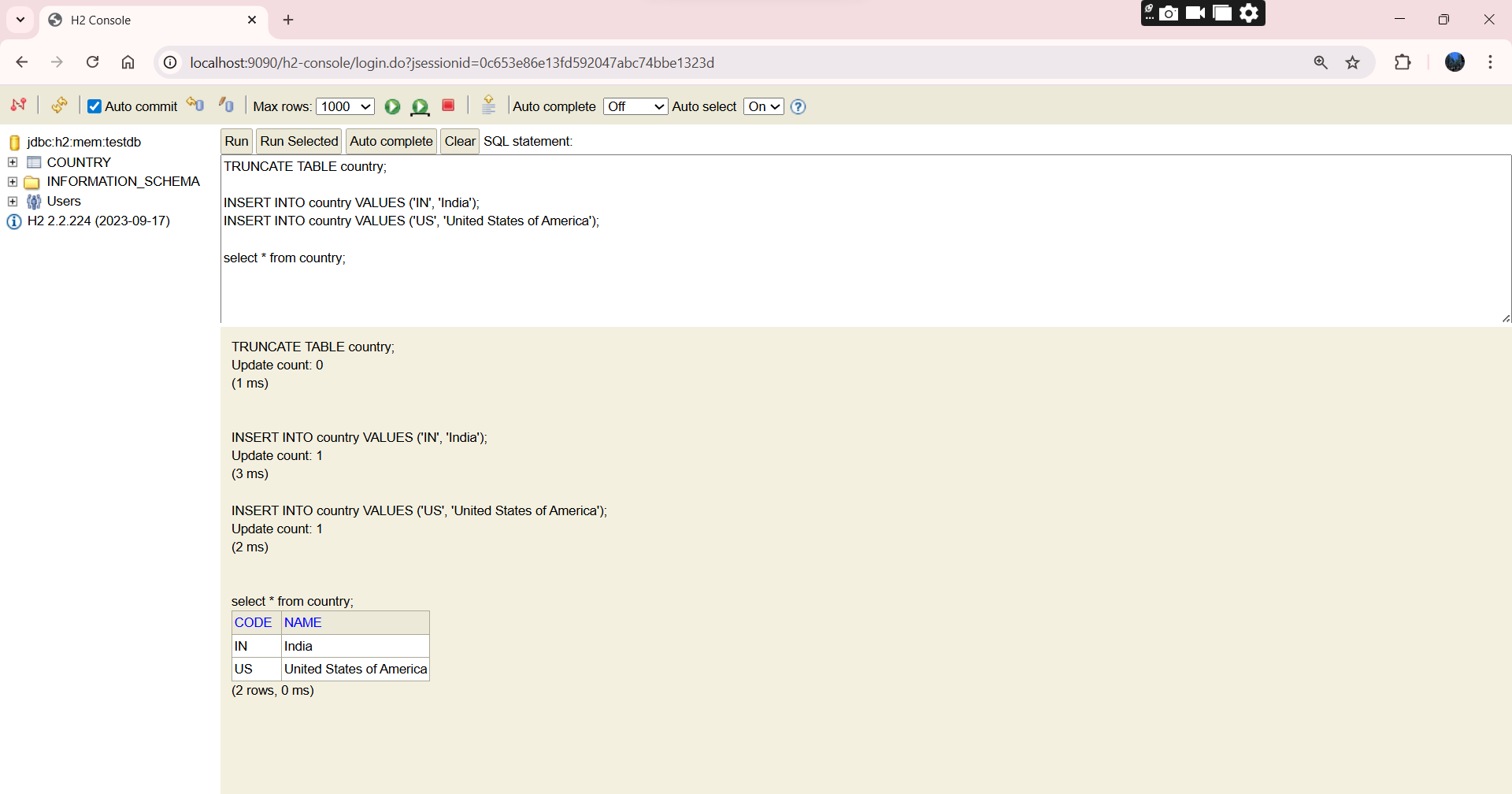
    </build>

</project>

**OUTPUT:**

**Initially:**



****

**(2). Hands on 4**

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

**Employee.java**

package com.example.demo;

import jakarta.persistence.\*;

@Entity

public class Employee {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Integer id;

    private String name;

    private String department;

    public Employee() {}

    public Employee(String name, String department) {

        this.name = name;

        this.department = department;

    }

    public Integer getId() {

        return id;

    }

    public String getName() {

        return name;

    }

    public String getDepartment() {

        return department;

    }

    public void setName(String name) {

        this.name = name;

    }

    public void setDepartment(String department) {

        this.department = department;

    }

}

**EmployeeRepository.java**

package com.example.demo;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

package com.example.demo;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class EmployeeService {

    @Autowired

    private EmployeeRepository employeeRepository;

    @Transactional

    public void addEmployee(Employee employee) {

        employeeRepository.save(employee);

    }

}

**DemoApplication.java**

package com.example.demo;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication implements CommandLineRunner {

    @Autowired

    private EmployeeService employeeService;

    public static void main(String[] args) {

        SpringApplication.run(DemoApplication.class, args);

    }

    @Override

    public void run(String... args) {

        Employee e1 = new Employee("Vanya", "Engineering");

        employeeService.addEmployee(e1);

        System.out.println("Employee saved using H2 Database!");

    }

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

server.port=9090

**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.example</groupId>

    <artifactId>spring-data-h2</artifactId>

    <version>1.0.0</version>

    <packaging>jar</packaging>

    <parent>

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

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

        <version>3.2.0</version>

    </parent>

    <dependencies>

        <!-- Spring Boot Web Starter -->

        <dependency>

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

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

        </dependency>

        <!-- Spring Data JPA -->

        <dependency>

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

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

        </dependency>

        <!-- H2 Database -->

        <dependency>

            <groupId>com.h2database</groupId>

            <artifactId>h2</artifactId>

            <scope>runtime</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

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

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

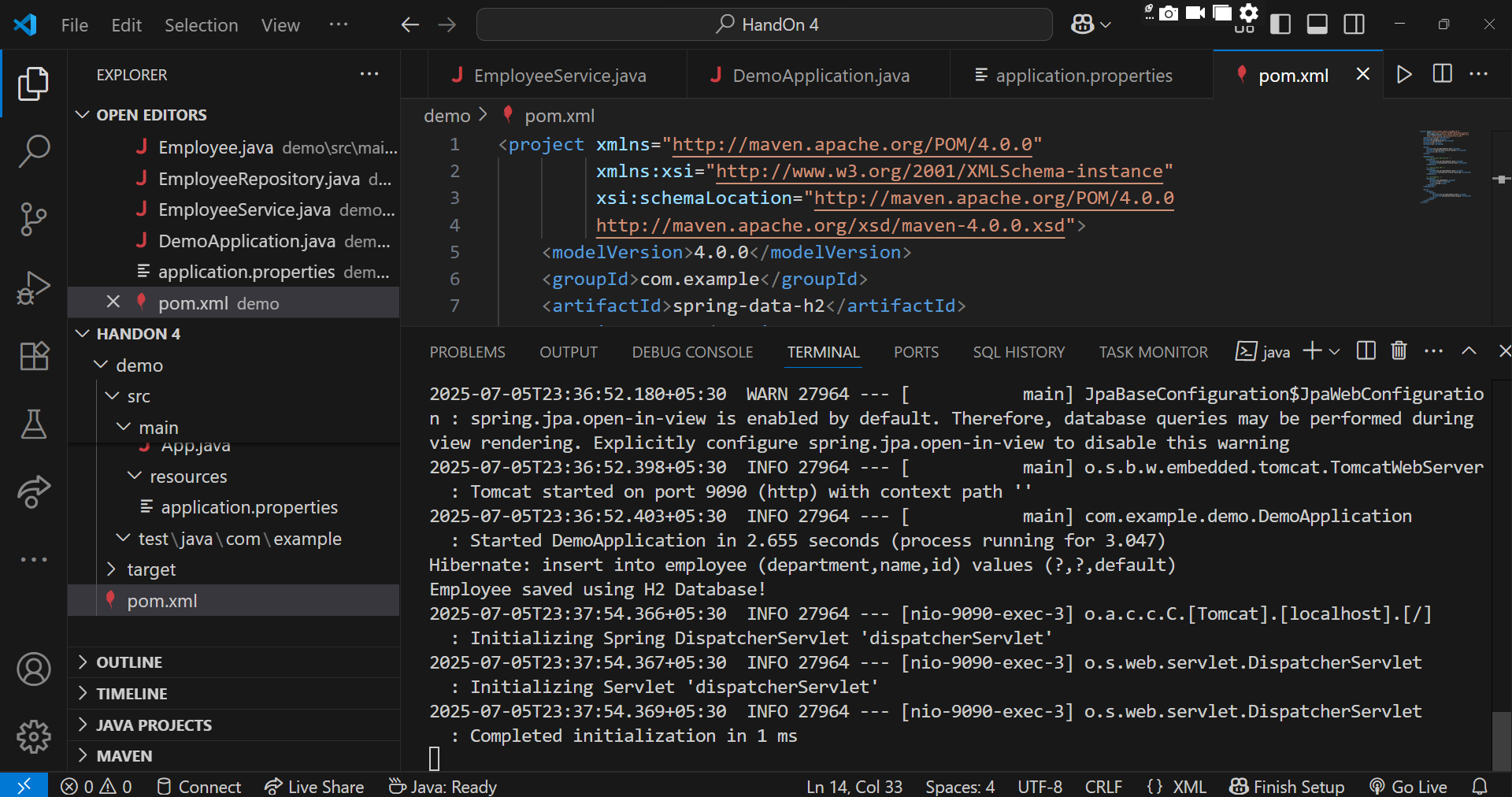
            </plugin>

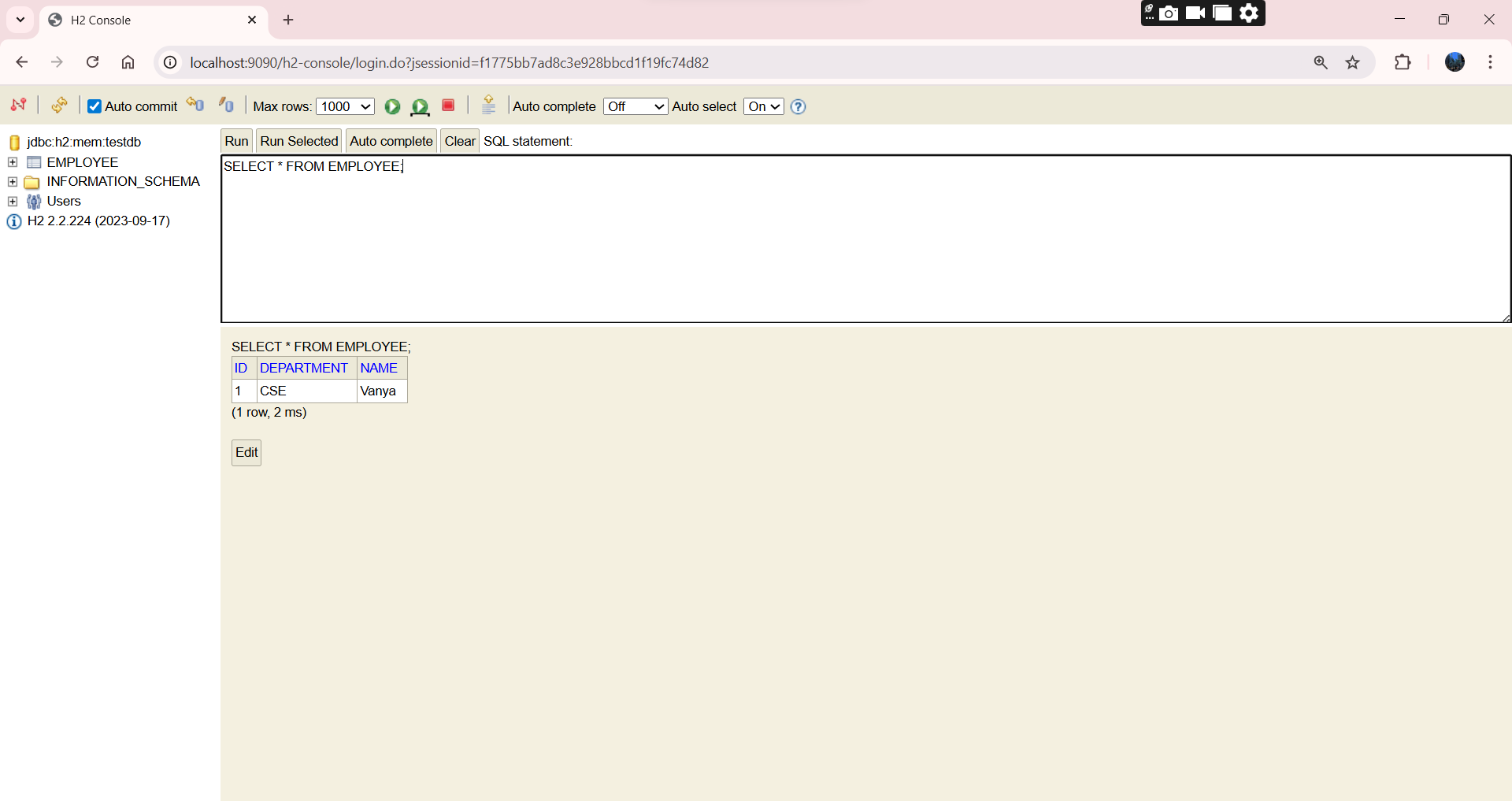
        </plugins>

    </build>

</project>

**OUTPUT:**

****

****

**Additional Important HandsOn**

**Exercise 5: Configuring the Spring IoC Container**

**Scenario:**

The library management application requires a central configuration for beans and dependencies.

**Steps:**

1. Create Spring Configuration File:
   * Create an XML configuration file named applicationContext.xml in the src/main/resources directory.
   * Define beans for BookService and BookRepository in the XML file.
2. Update the BookService Class:
   * Ensure that the BookService class has a setter method for BookRepository.
3. Run the Application:
   * Create a main class to load the Spring context and test the configuration.

Ans

**App.java**

package com.example.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.example.library.service.BookService;

public class App {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = context.getBean("bookService", BookService.class);

        bookService.addBook("Becoming");

    }

}

**BookService.java**

package com.example.library.service;

import com.example.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter for Dependency Injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook(String bookName) {

        System.out.println("Adding book: " + bookName);

        bookRepository.saveBook(bookName);

    }

}

**BookRepository.java**

package com.example.library.repository;

public class BookRepository {

    public void saveBook(String bookName) {

        System.out.println("Book '" + bookName + "' saved to the repository.");

    }

}

**applicationContext.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

       xsi:schemaLocation="http://www.springframework.org/schema/beans

                           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="bookRepository" class="com.example.library.repository.BookRepository" />

    <bean id="bookService" class="com.example.library.service.BookService">

        <property name="bookRepository" ref="bookRepository" />

    </bean>

</beans>

**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.example.librarymanagement</groupId>

    <artifactId>LibraryManagement</artifactId>

    <version>1.0-SNAPSHOT</version>

    <dependencies>

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

            <version>5.3.22</version>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.codehaus.mojo</groupId>

                <artifactId>exec-maven-plugin</artifactId>

                <version>3.5.1</version>

                <configuration>

                    <mainClass>com.example.library.App</mainClass>

                </configuration>

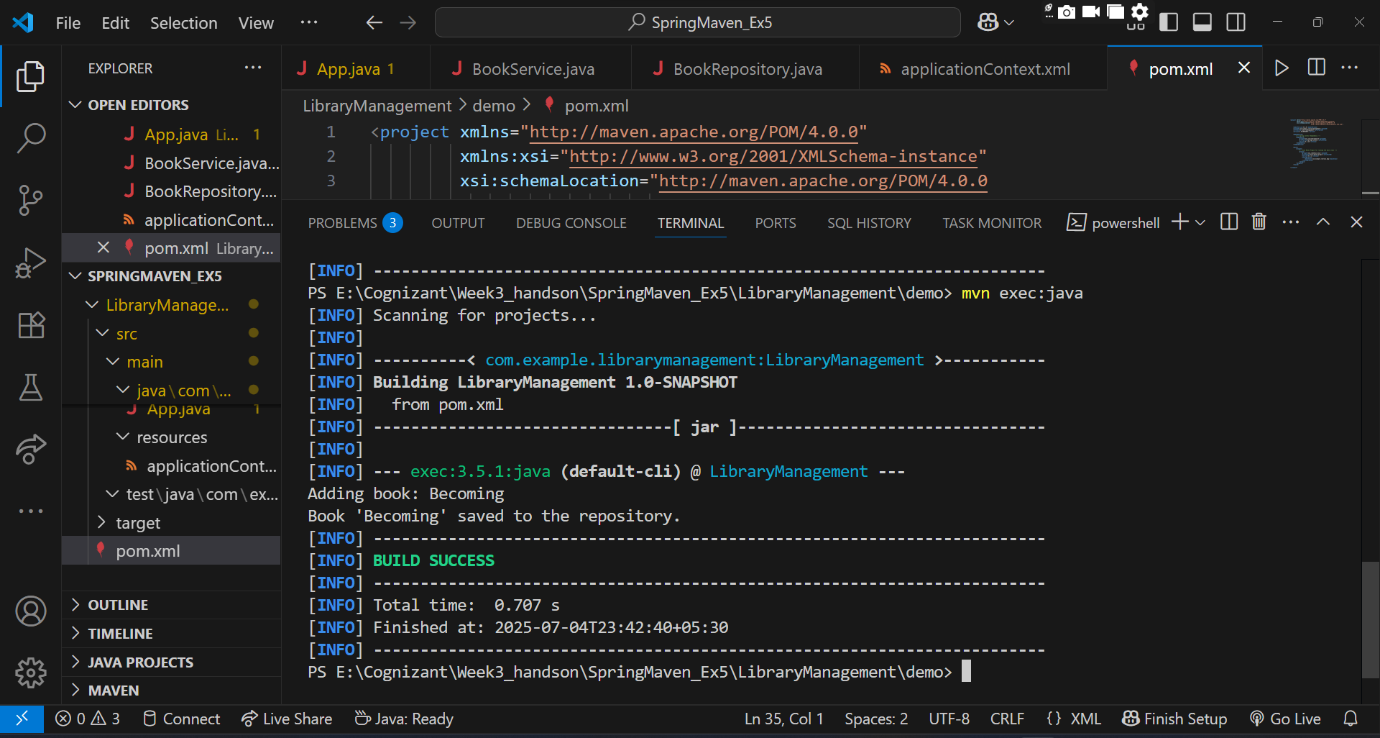
            </plugin>

        </plugins>

    </build>

</project>

**OUTPUT:**

****

**Exercise 7: Implementing Constructor and Setter Injection**

**Scenario:**

The library management application requires both constructor and setter injection for better control over bean initialization.

**Steps:**

1. Configure Constructor Injection:
   * Update applicationContext.xml to configure constructor injection for BookService.
2. Configure Setter Injection:
   * Ensure that the BookService class has a setter method for BookRepository and configure it in applicationContext.xml.
3. Test the Injection:
   * Run the LibraryManagementApplication main class to verify both constructor and setter injection.

Ans.

**LibraryManagementApplication.java**

package com.example.library;

import com.example.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = context.getBean("bookService", BookService.class);

        bookService.addBook();

    }

}

**Book.java**

package com.example.library.model;

public class Book {

    private String title;

    private String author;

    public Book() {

    }

    public Book(String title, String author) {

        this.title = title;

        this.author = author;

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public String getAuthor() {

        return author;

    }

    public void setAuthor(String author) {

        this.author = author;

    }

    @Override

    public String toString() {

        return "Book Title: " + title + ", Author: " + author;

    }

}

**BookRepository.java**

package com.example.library.repository;

public class BookRepository {

    public void save() {

        System.out.println("BookRepository: Book saved successfully to the database.");

    }

}

**BookService.java**

package com.example.library.service;

import com.example.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Constructor Injection

    public BookService(BookRepository bookRepository) {

        System.out.println("Constructor Injection: BookRepository has been injected.");

        this.bookRepository = bookRepository;

    }

    // Setter Injection

    public void setBookRepository(BookRepository bookRepository) {

        System.out.println("Setter Injection: BookRepository has been injected.");

        this.bookRepository = bookRepository;

    }

    public void addBook() {

        bookRepository.save();

        System.out.println("BookService: addBook() called.");

    }

    public BookRepository getBookRepository() {

        return bookRepository;

    }

}

**applicationContext.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

       xsi:schemaLocation="

           http://www.springframework.org/schema/beans

           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="bookRepository" class="com.example.library.repository.BookRepository" />

    <bean id="bookService" class="com.example.library.service.BookService">

        <constructor-arg ref="bookRepository" />

        <property name="bookRepository" ref="bookRepository" />

    </bean>

</beans>

**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.example.library</groupId>

    <artifactId>LibraryManagement</artifactId>

    <version>1.0</version>

    <dependencies>

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

            <version>5.3.29</version>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.codehaus.mojo</groupId>

                <artifactId>exec-maven-plugin</artifactId>

                <version>3.1.0</version>

                <configuration>

                    <mainClass>com.example.library.LibraryManagementApplication</mainClass>

                </configuration>

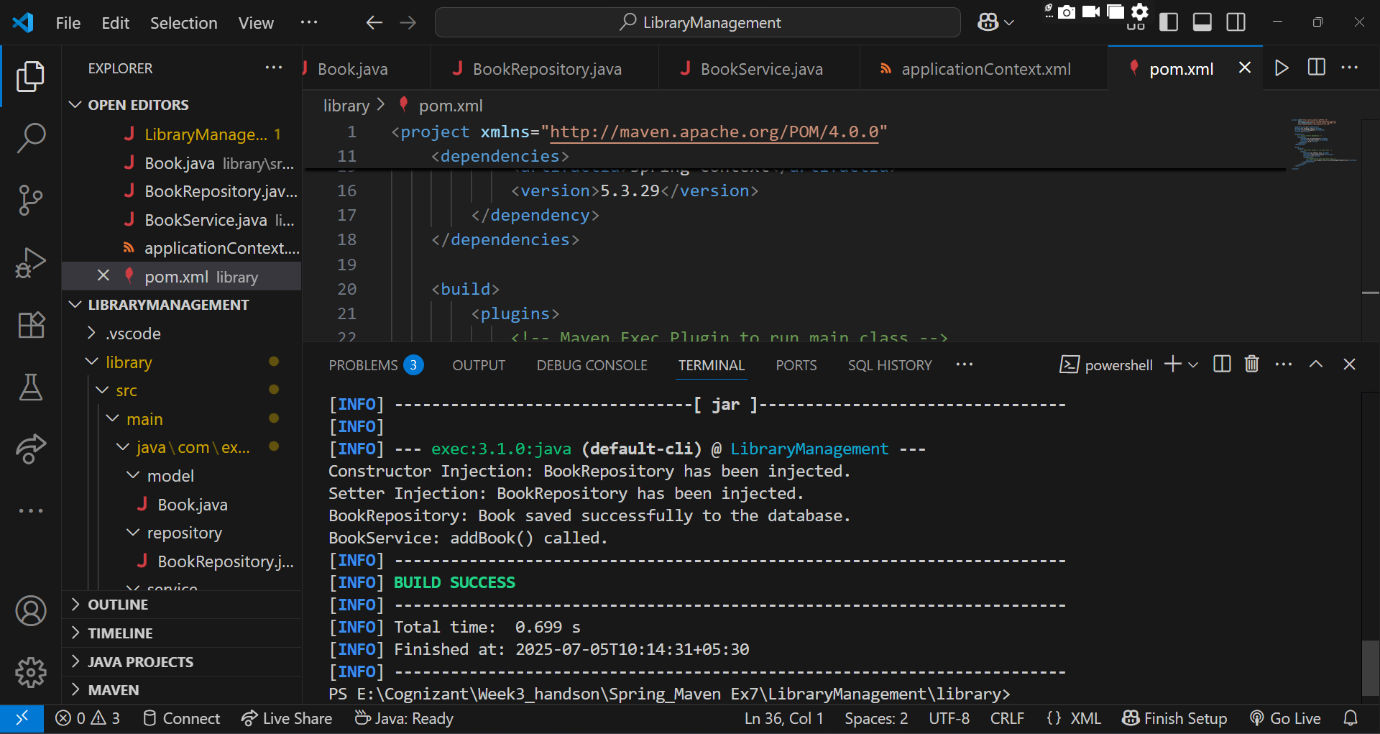
            </plugin>

        </plugins>

    </build>

</project>

**OUTPUT:**



**Exercise 9: Creating a Spring Boot Application**

**Scenario:**

You need to create a Spring Boot application for the library management system to simplify configuration and deployment.

**Steps:**

1. Create a Spring Boot Project:
   * Use Spring Initializr to create a new Spring Boot project named LibraryManagement.
2. Add Dependencies:
   * Include dependencies for Spring Web, Spring Data JPA, and H2 Database.
3. Create Application Properties:
   * Configure database connection properties in application.properties.
4. Define Entities and Repositories:
   * Create Book entity and BookRepository interface.
5. Create a REST Controller:
   * Create a BookController class to handle CRUD operations.
6. Run the Application:
   * Run the Spring Boot application and test the REST endpoints.

**Ans.**

**LibraryManagementApplication.java**

package com.example.librarymanagement;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibraryManagementApplication {

    public static void main(String[] args) {

        SpringApplication.run(LibraryManagementApplication.class, args);

    }

}

**Book.java**

package com.example.librarymanagement.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Book {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String title;

    private String author;

    private String category;

    public Book() {}

    public Book(String title, String author, String category) {

        this.title = title;

        this.author = author;

        this.category = category;

    }

    public Long getId() { return id; }

    public void setId(Long id) { this.id = id; }

    public String getTitle() { return title; }

    public void setTitle(String title) { this.title = title; }

    public String getAuthor() { return author; }

    public void setAuthor(String author) { this.author = author; }

    public String getCategory() { return category; }

    public void setCategory(String category) { this.category = category; }

}

**BookController.java**

package com.example.librarymanagement.controller;

import com.example.librarymanagement.entity.Book;

import com.example.librarymanagement.repository.BookRepository;

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

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

    @Autowired

    private BookRepository bookRepository;

    @GetMapping

    public List<Book> getAllBooks() {

        return bookRepository.findAll();

    }

    @GetMapping("/{id}")

    public Book getBookById(@PathVariable Long id) {

        return bookRepository.findById(id).orElse(null);

    }

    @PostMapping

    public Book addBook(@RequestBody Book book) {

        return bookRepository.save(book);

    }

    @PutMapping("/{id}")

    public Book updateBook(@PathVariable Long id, @RequestBody Book updatedBook) {

        return bookRepository.findById(id).map(book -> {

            book.setTitle(updatedBook.getTitle());

            book.setAuthor(updatedBook.getAuthor());

            book.setCategory(updatedBook.getCategory());

            return bookRepository.save(book);

        }).orElse(null);

    }

    @DeleteMapping("/{id}")

    public void deleteBook(@PathVariable Long id) {

        bookRepository.deleteById(id);

    }

}

**BookRepository.java**

package com.example.librarymanagement.repository;

import com.example.librarymanagement.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**application.properties**

server.port=9090

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

spring.datasource.url=jdbc:h2:mem:librarydb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

**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

                             http://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>3.2.5</version>

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

  </parent>

  <groupId>com.example</groupId>

  <artifactId>LibraryManagement</artifactId>

  <version>1.0</version>

  <name>LibraryManagement</name>

  <description>Spring Boot Library Management System</description>

  <properties>

    <java.version>17</java.version>

  </properties>

  <dependencies>

    <!-- Spring Boot Web for REST APIs -->

    <dependency>

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

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

    </dependency>

    <!-- Spring Data JPA -->

    <dependency>

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

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

    </dependency>

    <!-- H2 in-memory database -->

    <dependency>

      <groupId>com.h2database</groupId>

      <artifactId>h2</artifactId>

      <scope>runtime</scope>

    </dependency>

    <!-- Optional: For testing -->

    <dependency>

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

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

      <scope>test</scope>

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <!-- Spring Boot Plugin -->

      <plugin>

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

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

        <version>3.2.5</version>

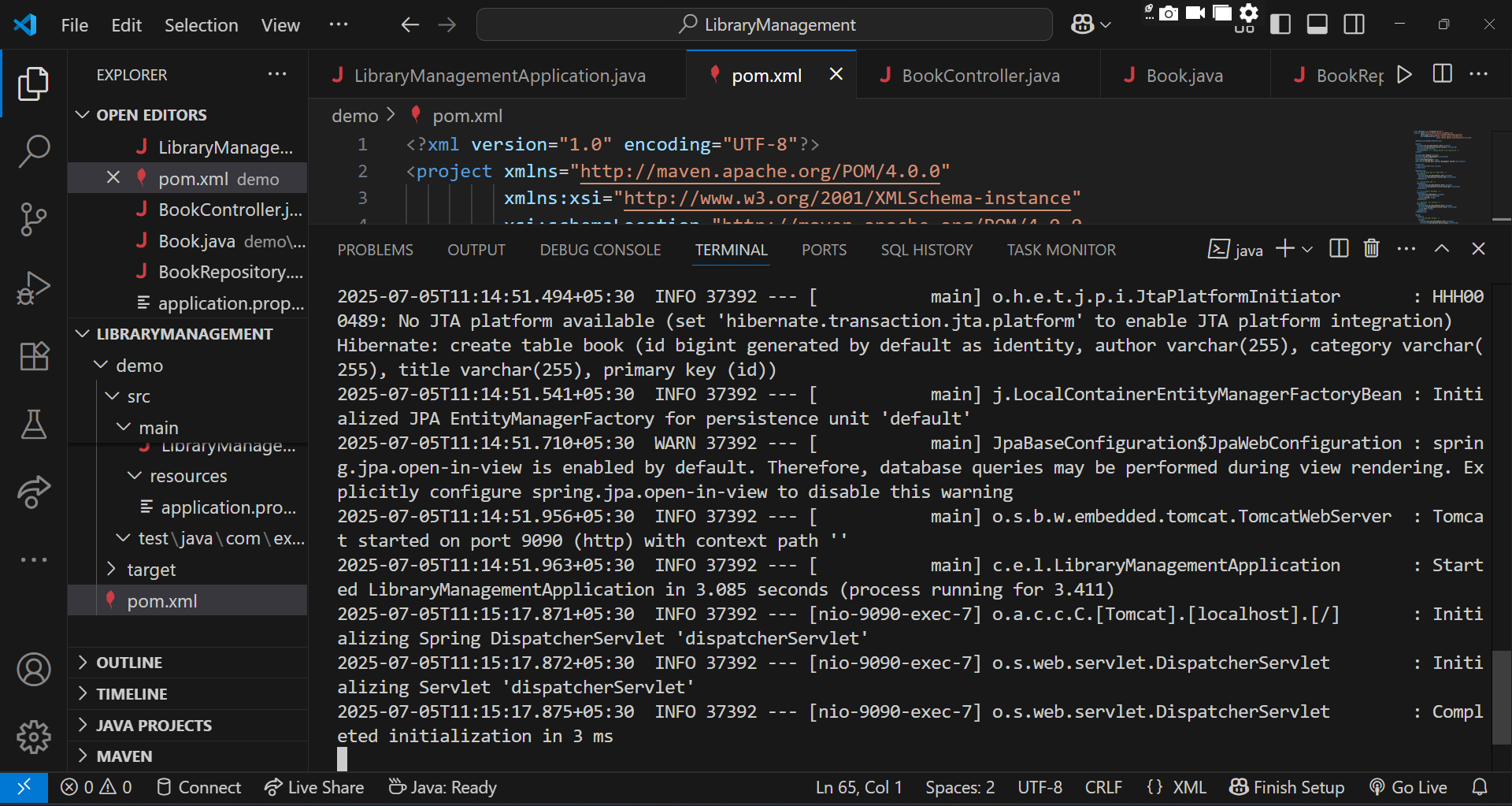
      </plugin>

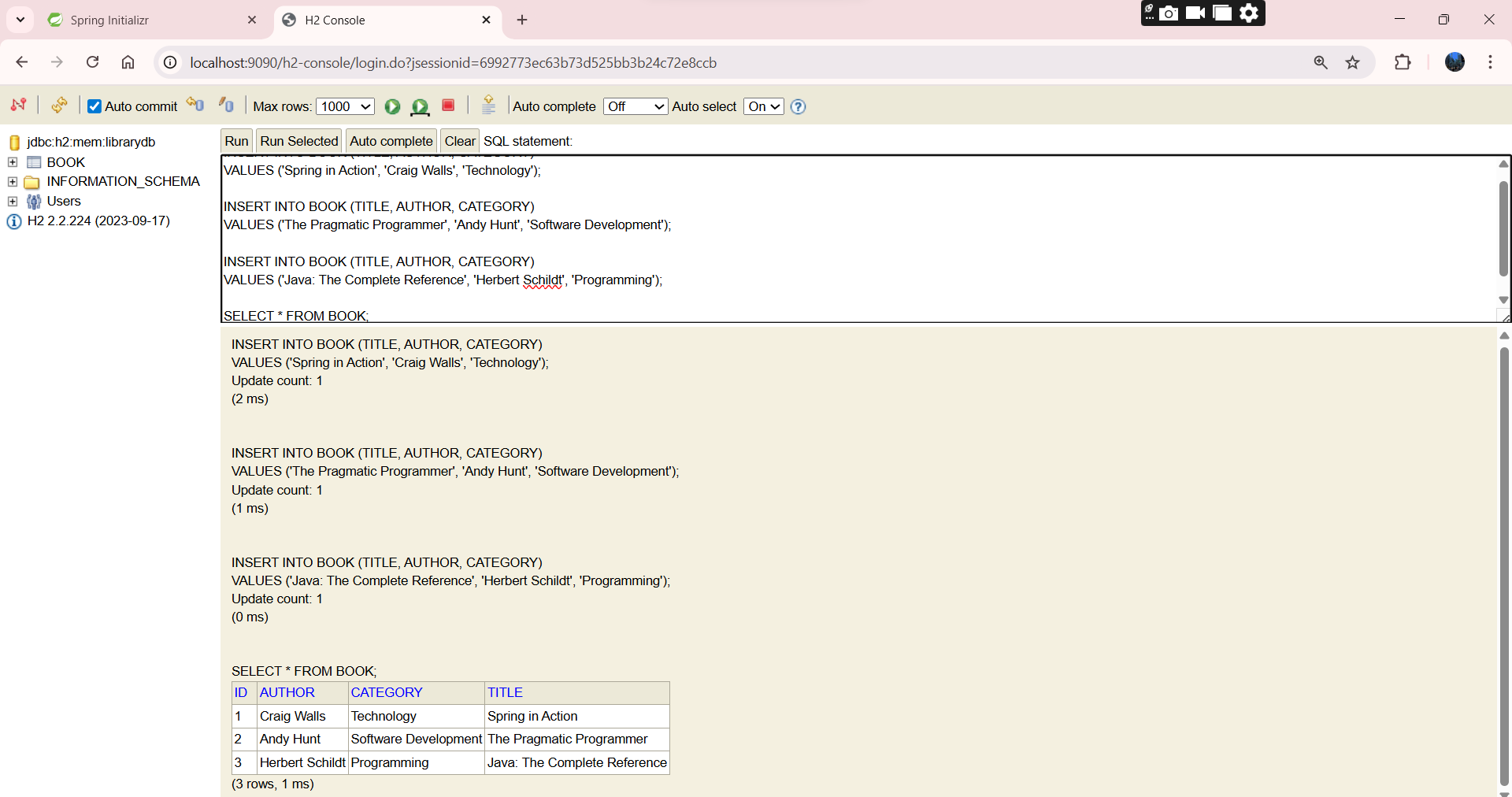
    </plugins>

  </build>

</project>

**OUTPUT:**





**1.SpringData-JPA-Handson**

**Hands on 5**

**Implement services for managing Country**

**SpringLearnCountryLookupApplication.java**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringlearnCountryLookupApplication {

    public static void main(String[] args) {

        SpringApplication.run(SpringlearnCountryLookupApplication.class, args);

    }

}

**Country.java**

package com.cognizant.springlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

    @Id

    private String coCode;

    private String coName;

    public Country() {}

    public Country(String coCode, String coName) {

        this.coCode = coCode;

        this.coName = coName;

    }

    public String getCoCode() { return coCode; }

    public void setCoCode(String coCode) { this.coCode = coCode; }

    public String getCoName() { return coName; }

    public void setCoName(String coName) { this.coName = coName; }

}

**CountryController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

    @Autowired

    private CountryService service;

    @GetMapping("/{code}")

    public Country getCountry(@PathVariable String code) throws CountryNotFoundException {

        return service.getCountry(code);

    }

    @PostMapping

    public Country addCountry(@RequestBody Country country) {

        return service.addCountry(country);

    }

    @PutMapping

    public Country updateCountry(@RequestBody Country country) throws CountryNotFoundException {

        return service.updateCountry(country);

    }

    @DeleteMapping("/{code}")

    public void deleteCountry(@PathVariable String code) throws CountryNotFoundException {

        service.deleteCountry(code);

    }

    @GetMapping("/search")

    public List<Country> searchCountries(@RequestParam String namePart) {

        return service.searchByName(namePart);

    }

    @ExceptionHandler(CountryNotFoundException.class)

    public ResponseEntity<String> handleCountryNotFoundException(CountryNotFoundException ex) {

        return new ResponseEntity<>(ex.getMessage(), HttpStatus.NOT\_FOUND);

    }

}

**CountryService.java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import java.util.List;

public interface CountryService {

    Country getCountry(String code) throws CountryNotFoundException;

    Country addCountry(Country country);

    Country updateCountry(Country country) throws CountryNotFoundException;

    void deleteCountry(String code) throws CountryNotFoundException;

    List<Country> searchByName(String namePart);

}

**CountryServiceImpl.java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.repository.CountryRepository;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryServiceImpl implements CountryService {

    @Autowired

    private CountryRepository repository;

    @Override

    public Country getCountry(String code) throws CountryNotFoundException {

        return repository.findById(code).orElseThrow(() ->

            new CountryNotFoundException("Country code not found: " + code));

    }

    @Override

    public Country addCountry(Country country) {

        return repository.save(country);

    }

    @Override

    public Country updateCountry(Country country) throws CountryNotFoundException {

        if (!repository.existsById(country.getCoCode())) {

            throw new CountryNotFoundException("Cannot update: Code not found");

        }

        return repository.save(country);

    }

    @Override

    public void deleteCountry(String code) throws CountryNotFoundException {

        if (!repository.existsById(code)) {

            throw new CountryNotFoundException("Cannot delete: Code not found");

        }

        repository.deleteById(code);

    }

    @Override

    public List<Country> searchByName(String namePart) {

        return repository.findByCoNameContainingIgnoreCase(namePart);

    }

}

**CountryNotFoundException.java**

package com.cognizant.springlearn.service.exception;

public class CountryNotFoundException extends Exception {

    public CountryNotFoundException(String message) {

        super(message);

    }

}

**CountryRepository.java**

package com.cognizant.springlearn.repository;

import com.cognizant.springlearn.model.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

    List<Country> findByCoNameContainingIgnoreCase(String namePart);

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

server.port=8081

**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>springlearn-country-lookup</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <packaging>jar</packaging>

    <name>springlearn-country-lookup</name>

    <description>Spring Boot Country Lookup with H2</description>

    <parent>

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

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

        <version>3.2.4</version>

        <relativePath/>

    </parent>

    <properties>

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

    </properties>

    <dependencies>

        <dependency>

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

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

        </dependency>

        <dependency>

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

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

        </dependency>

        <dependency>

            <groupId>com.h2database</groupId>

            <artifactId>h2</artifactId>

            <scope>runtime</scope>

        </dependency>

        <dependency>

            <groupId>jakarta.persistence</groupId>

            <artifactId>jakarta.persistence-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.springframework.boot</groupId>

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

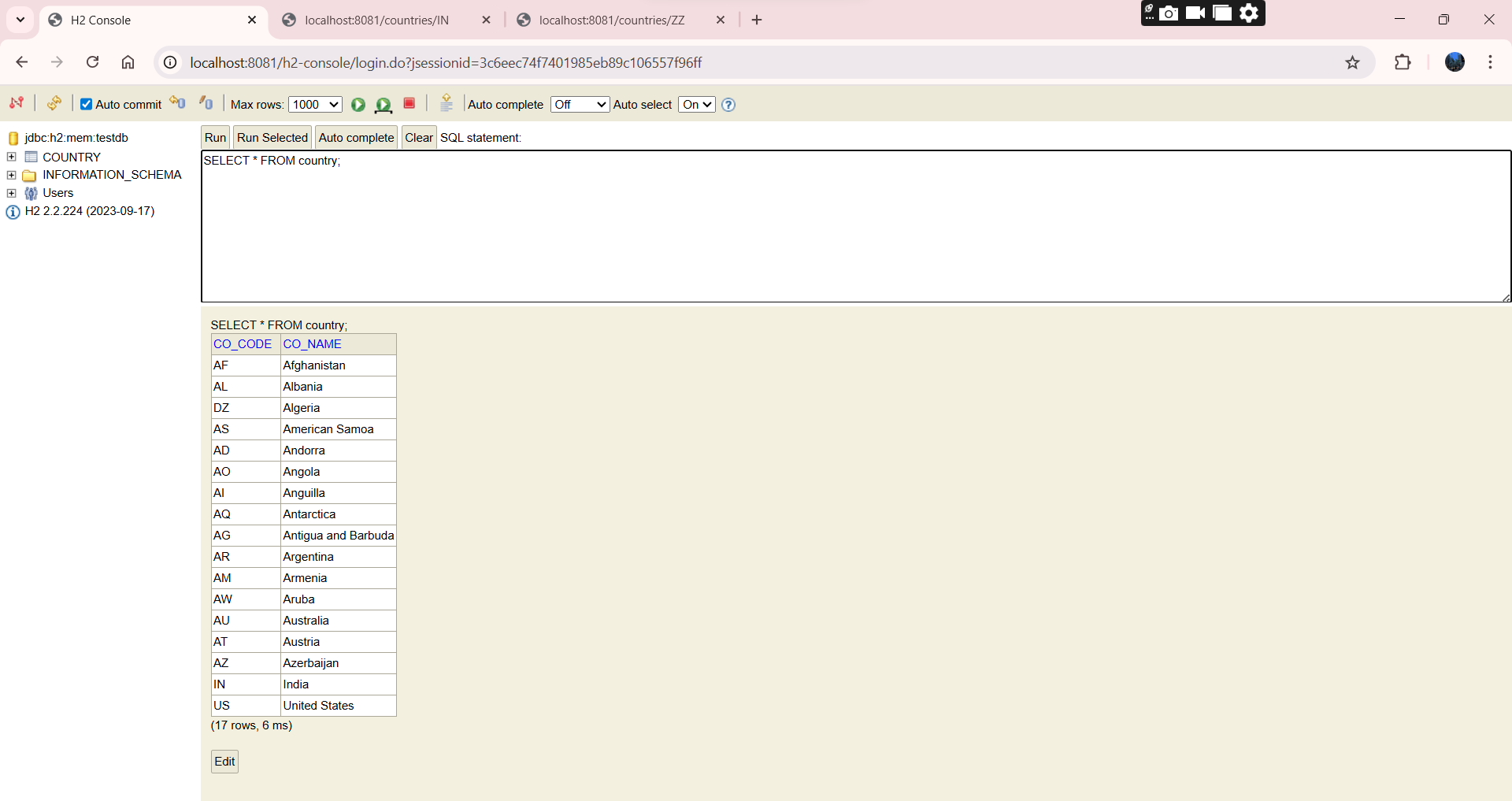
            </plugin>

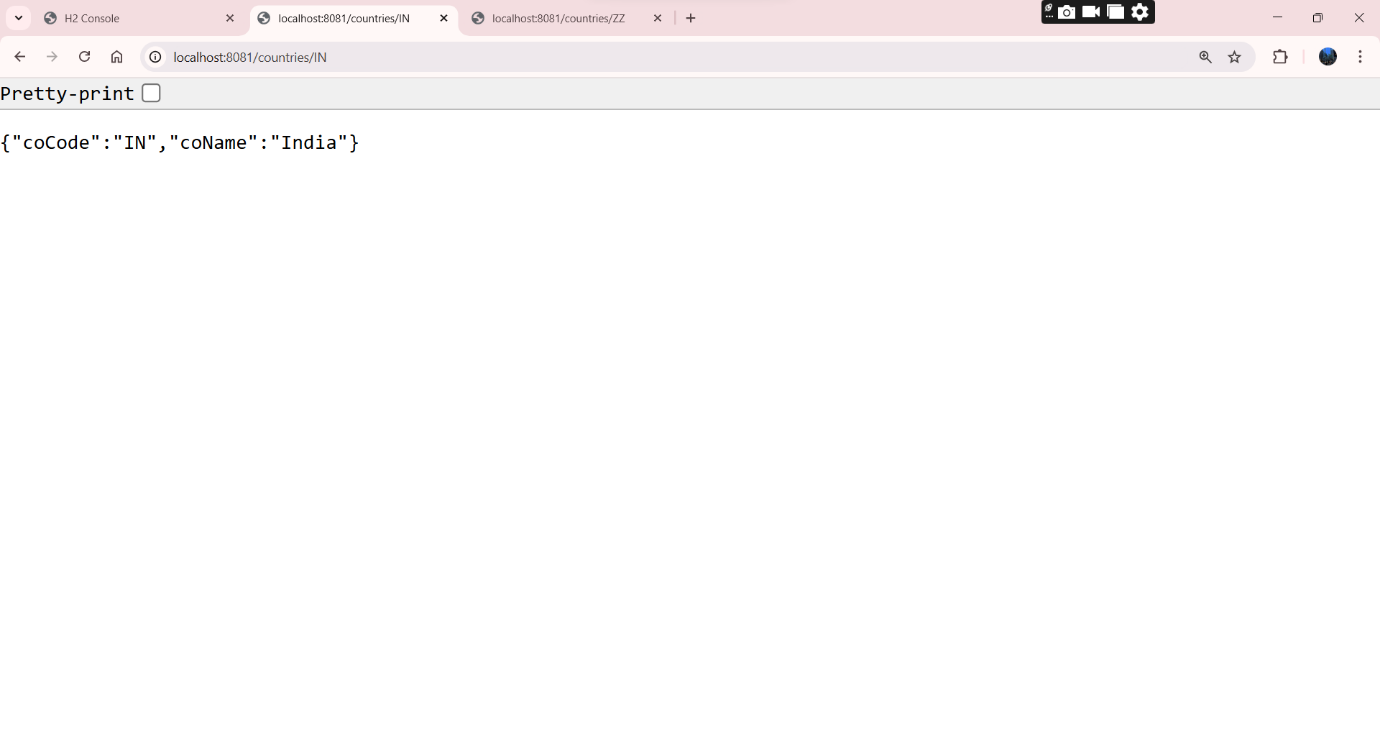
        </plugins>

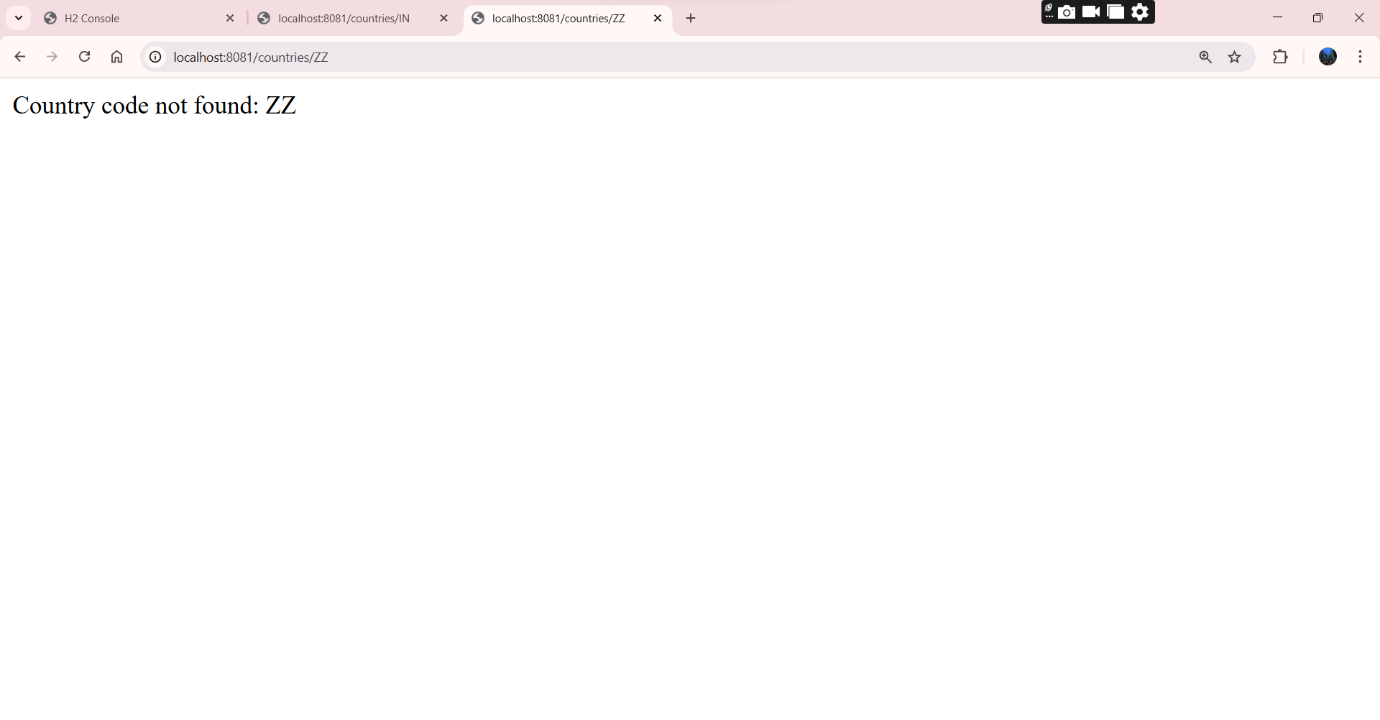
    </build>

</project>

**OUTPUT:**







**Hands on 6**

**Find a country based on country code**

Ans.

**CountryController.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

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

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/countries")

public class CountryController {

    @Autowired

    private CountryService service;

    @GetMapping("/{code}")

    public Country getCountry(@PathVariable String code) throws CountryNotFoundException {

        return service.findCountryByCode(code);

    }

}

**Country.java**

package com.cognizant.springlearn.model;

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.springlearn.repository;

import com.cognizant.springlearn.model.Country;

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

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

**CountryService.java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import java.util.List;

public interface CountryService {

    Country findCountryByCode(String countryCode) throws CountryNotFoundException;

    List<Country> getAllCountries();

}

**CountryServiceImpl.java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.repository.CountryRepository;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class CountryServiceImpl implements CountryService {

    @Autowired

    private CountryRepository countryRepository;

    @Override

    @Transactional

    public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

        Optional<Country> result = countryRepository.findById(countryCode);

        if (result.isEmpty()) {

            throw new CountryNotFoundException("Country code not found: " + countryCode);

        }

        return result.get();

    }

    @Override

    public List<Country> getAllCountries() {

        return countryRepository.findAll();

    }

}

**OrmLearnApplication.java**

package com.cognizant.springlearn;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan("com.cognizant.springlearn")

public class OrmLearnApplication implements CommandLineRunner {

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

    @Autowired

    private CountryService countryService;

    public static void main(String[] args) {

        SpringApplication.run(OrmLearnApplication.class, args);

    }

    @Override

    public void run(String... args) {

        LOGGER.info("Start");

        try {

            Country country = countryService.findCountryByCode("IN");

            LOGGER.info("Country: {} - {}", country.getCode(), country.getName());

        } catch (CountryNotFoundException e) {

            LOGGER.error("Exception: {}", e.getMessage());

        }

        LOGGER.info("End");

    }

}

**CountryNotFoundException.java**

package com.cognizant.springlearn.service.exception;

public class CountryNotFoundException extends Exception {

    public CountryNotFoundException(String message) {

        super(message);

    }

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.h2.console.enabled=true

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

server.port=8081

**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

         http://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>3.2.4</version>

        <relativePath/>

    </parent>

    <groupId>com.cognizant</groupId>

    <artifactId>springlearn-country-lookup</artifactId>

    <version>1.0</version>

    <name>springlearn-country-lookup</name>

    <packaging>jar</packaging>

    <properties>

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

    </properties>

    <dependencies>

        <dependency>

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

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

        </dependency>

        <dependency>

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

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

        </dependency>

        <dependency>

            <groupId>com.h2database</groupId>

            <artifactId>h2</artifactId>

            <scope>runtime</scope>

        </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>

            <version>3.2.4</version>

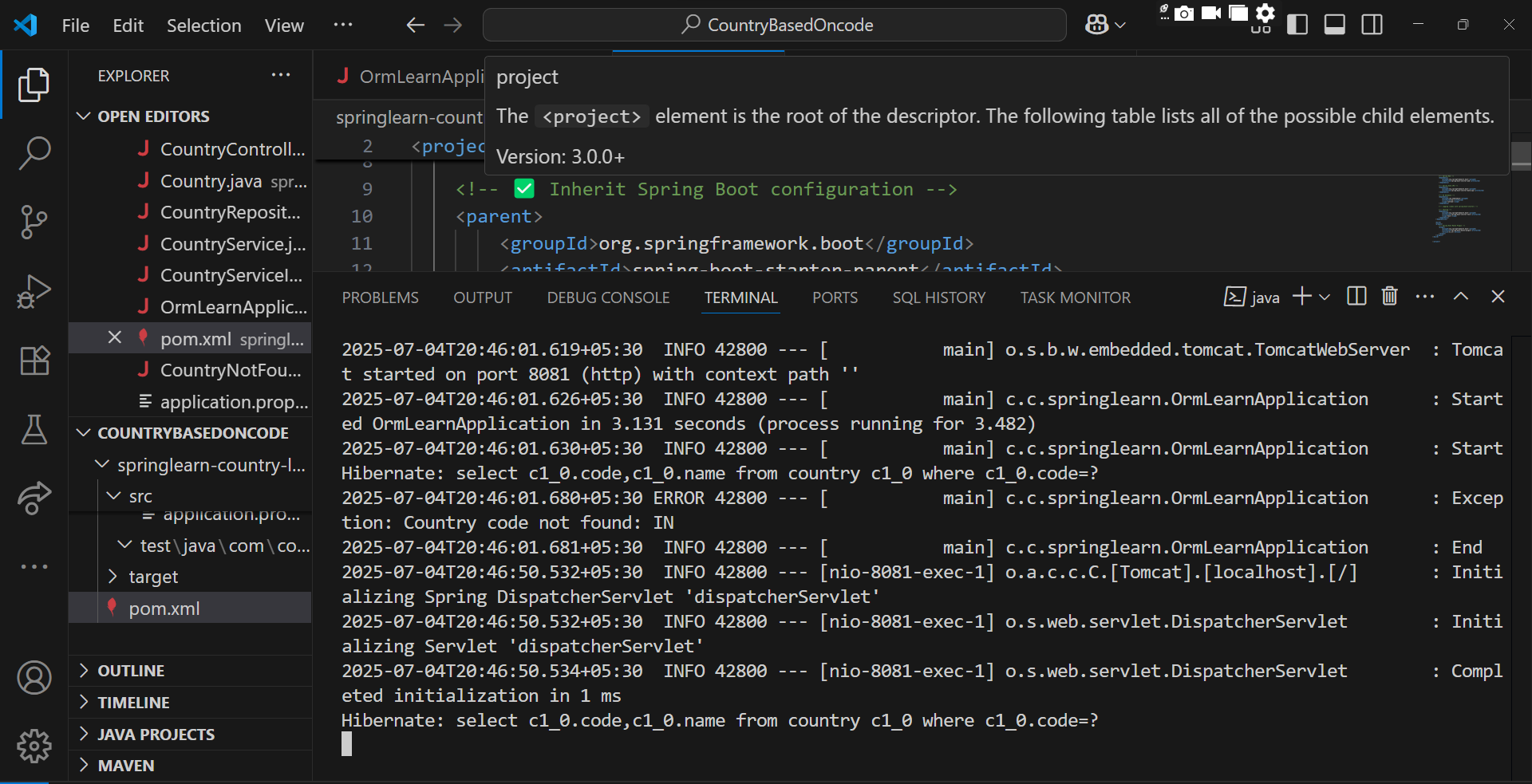
        </plugin>

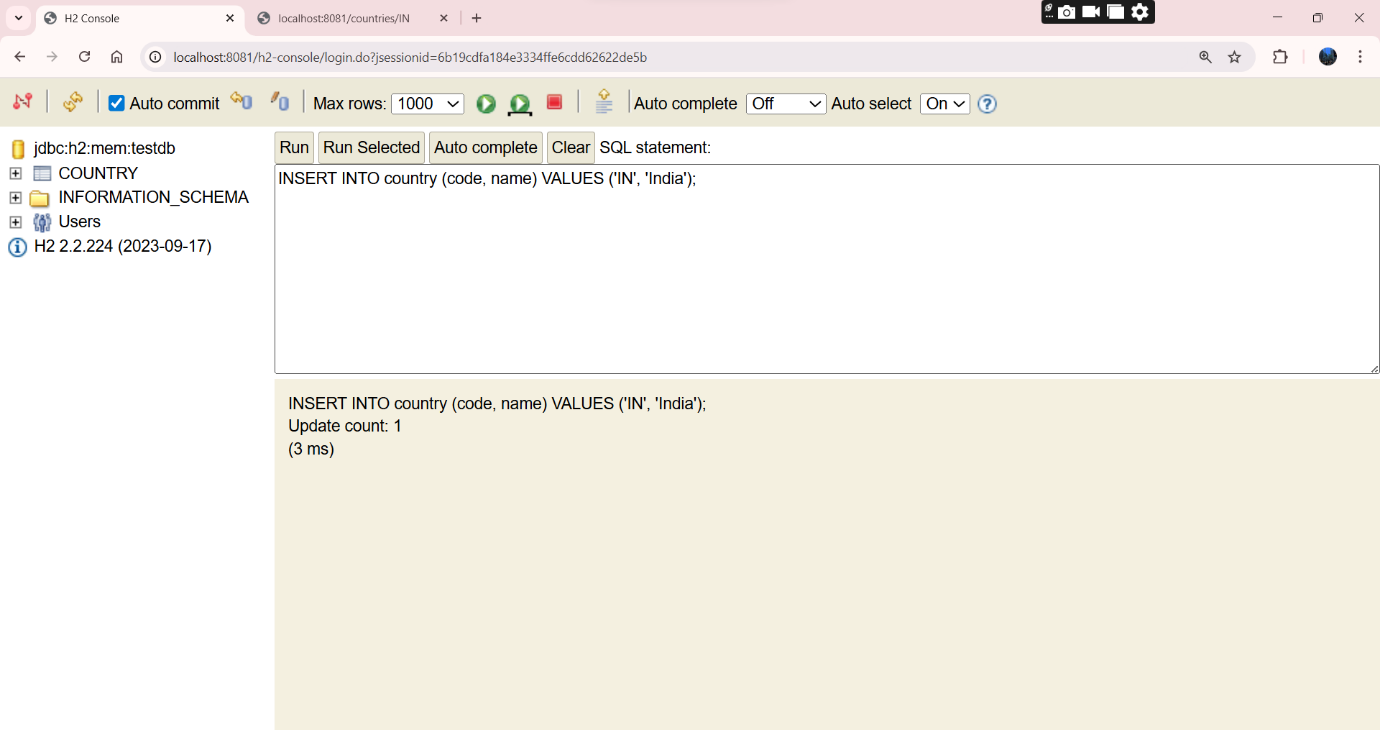
    </plugins>

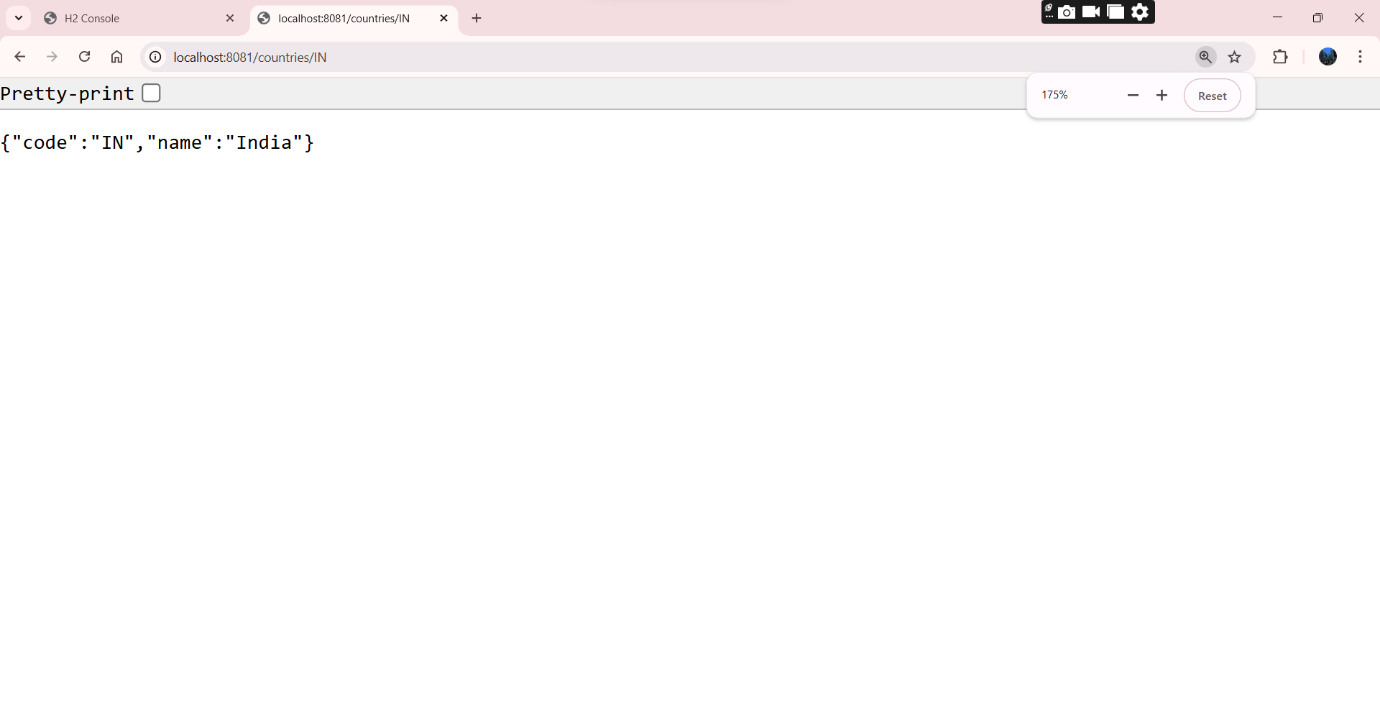
</build>

</project>

**OUTPUT:**







**Hands on 7**

**Add a new country**

**OrmLearnApplication.java**

package com.cognizant.springlearn;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

@SpringBootApplication

public class OrmLearnApplication {

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

    public static void main(String[] args) {

        ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

        LOGGER.info("Start");

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

        Country newCountry = new Country("JP", "Japan");

        countryService.addCountry(newCountry);

        try {

            Country retrieved = countryService.findCountryByCode("JP");

            System.out.println("Added and Retrieved Country: " + retrieved);

        } catch (CountryNotFoundException e) {

            LOGGER.error("Exception: " + e.getMessage());

        }

        LOGGER.info("End");

    }

}

**Country.java**

package com.cognizant.springlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

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.springlearn.repository;

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

import com.cognizant.springlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**

package com.cognizant.springlearn.service;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.repository.CountryRepository;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

@Service

public class CountryService {

    private final CountryRepository countryRepository;

    public CountryService(CountryRepository countryRepository) {

        this.countryRepository = countryRepository;

    }

    @Transactional

    public Country findCountryByCode(String code) throws CountryNotFoundException {

        return countryRepository.findById(code)

                .orElseThrow(() -> new CountryNotFoundException("Country code not found: " + code));

    }

    @Transactional

    public void addCountry(Country country) {

        countryRepository.save(country);

    }

}

**CountryNotFoundException.java**

package com.cognizant.springlearn.service.exception;

public class CountryNotFoundException extends Exception {

    public CountryNotFoundException(String message) {

        super(message);

    }

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=create

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

server.port=8081

**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

                             http://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>3.2.5</version>

    <relativePath/>

  </parent>

  <groupId>com.cognizant</groupId>

  <artifactId>springlearn-country-lookup</artifactId>

  <version>1.0</version>

  <name>springlearn-country-lookup</name>

  <properties>

    <java.version>17</java.version>

  </properties>

  <dependencies>

  <!-- Spring Boot Core -->

  <dependency>

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

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

  </dependency>

  <dependency>

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

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

  </dependency>

  <dependency>

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

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

  </dependency>

  <dependency>

    <groupId>com.h2database</groupId>

    <artifactId>h2</artifactId>

    <scope>runtime</scope>

  </dependency>

  <dependency>

    <groupId>org.junit.jupiter</groupId>

    <artifactId>junit-jupiter</artifactId>

    <scope>test</scope>

  </dependency>

</dependencies>

  <build>

    <plugins>

      <plugin>

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

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

      </plugin>

      <plugin>

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

        <version>3.10.1</version>

        <configuration>

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

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

        </configuration>

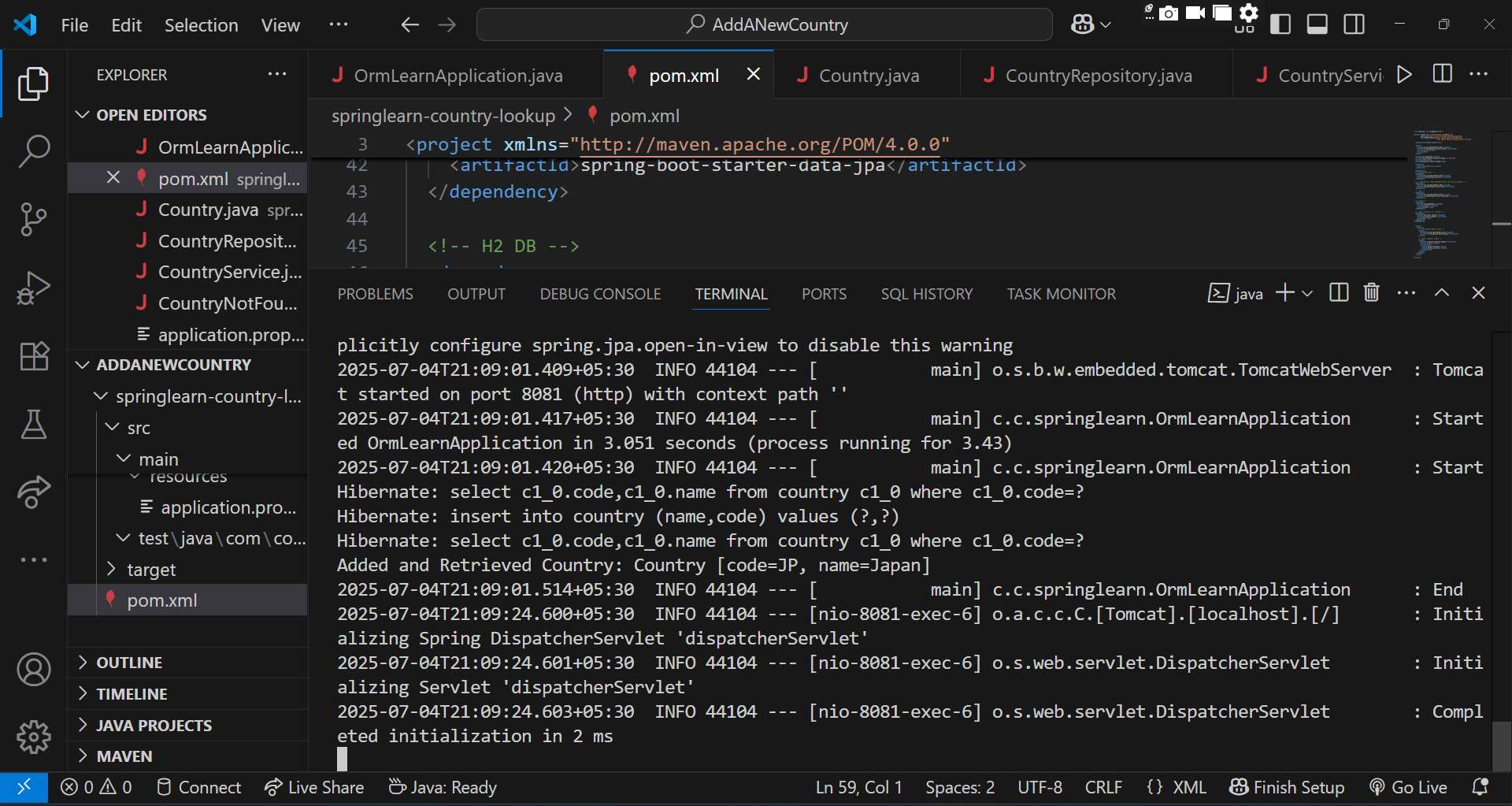
      </plugin>

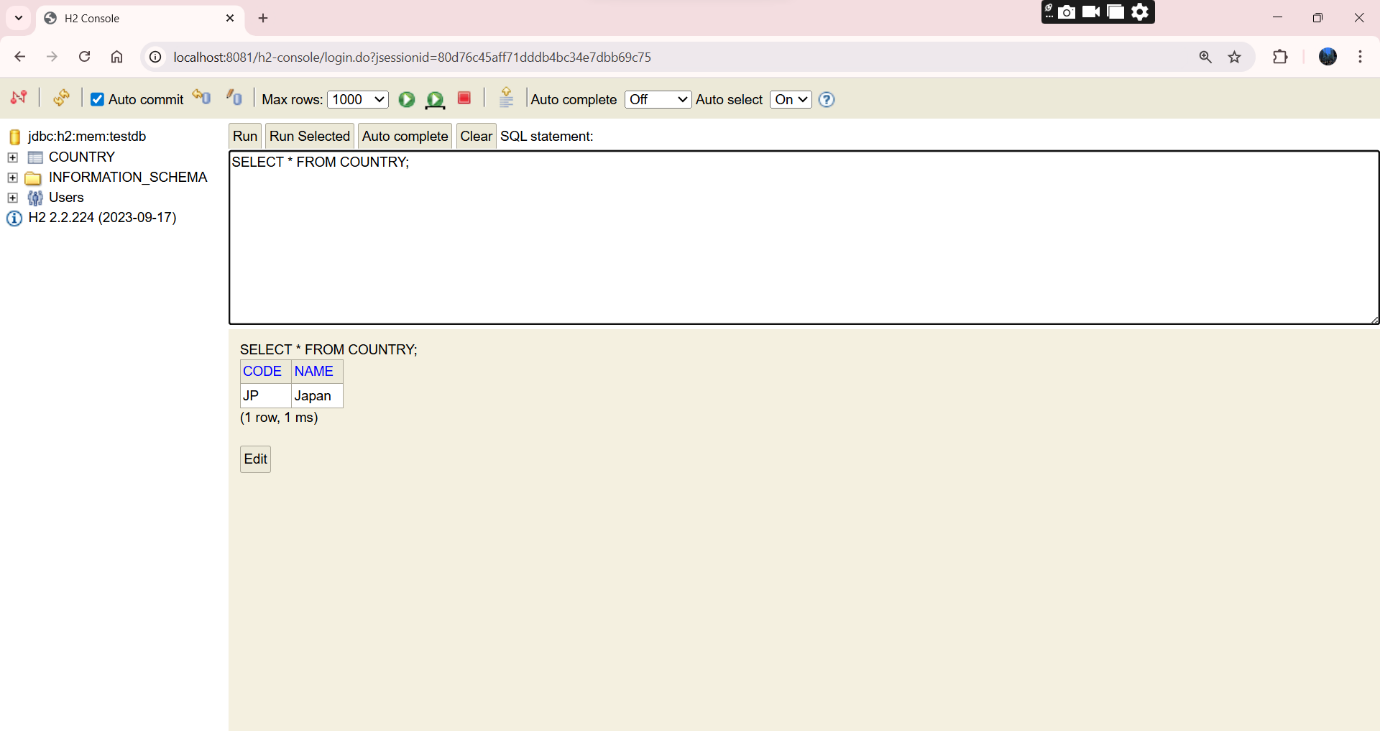
    </plugins>

  </build>

</project>

**OUTPUT:**





**2.Spring-Data-jpa-Handson**

**Objectives**

* Demonstrate implementation of Query Methods feature of Spring Data JPA

Ans

**Country.java**

package com.cognizant.springlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import java.time.LocalDate;

@Entity

public class Country {

    @Id

    private String code;

    private String name;

    private long population;

    private LocalDate independenceDate;

    public Country() {}

    public Country(String code, String name, long population, LocalDate independenceDate) {

        this.code = code;

        this.name = name;

        this.population = population;

        this.independenceDate = independenceDate;

    }

    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; }

    public long getPopulation() { return population; }

    public void setPopulation(long population) { this.population = population; }

    public LocalDate getIndependenceDate() { return independenceDate; }

    public void setIndependenceDate(LocalDate independenceDate) { this.independenceDate = independenceDate; }

    @Override

    public String toString() {

        return "Country [code=" + code + ", name=" + name + ", population=" + population +

                ", independenceDate=" + independenceDate + "]";

    }

}

**CountryRepository.java**

package com.cognizant.springlearn.repository;

import com.cognizant.springlearn.model.Country;

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

import java.time.LocalDate;

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

    List<Country> findByNameContaining(String text);

    List<Country> findByNameStartingWith(String prefix);

    List<Country> findByIndependenceDateBetween(LocalDate start, LocalDate end);

    List<Country> findByPopulationGreaterThan(long population);

    List<Country> findByPopulationLessThan(long population);

    List<Country> findTop3ByOrderByPopulationDesc();

    List<Country> findAllByOrderByNameAsc();

}

**CountryService.java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.repository.CountryRepository;

import jakarta.transaction.Transactional;

import org.springframework.stereotype.Service;

import java.time.LocalDate;

import java.util.List;

@Service

public class CountryService {

    private final CountryRepository countryRepository;

    public CountryService(CountryRepository countryRepository) {

        this.countryRepository = countryRepository;

    }

    @Transactional

    public void addCountry(Country country) {

        countryRepository.save(country);

    }

    public List<Country> findByNameContaining(String text) {

        return countryRepository.findByNameContaining(text);

    }

    public List<Country> findByNameStartingWith(String prefix) {

        return countryRepository.findByNameStartingWith(prefix);

    }

    public List<Country> findByIndependenceDateBetween(LocalDate start, LocalDate end) {

        return countryRepository.findByIndependenceDateBetween(start, end);

    }

    public List<Country> findByPopulationGreaterThan(long population) {

        return countryRepository.findByPopulationGreaterThan(population);

    }

    public List<Country> findByPopulationLessThan(long population) {

        return countryRepository.findByPopulationLessThan(population);

    }

    public List<Country> findTop3ByOrderByPopulationDesc() {

        return countryRepository.findTop3ByOrderByPopulationDesc();

    }

    public List<Country> findAllSortedByName() {

        return countryRepository.findAllByOrderByNameAsc();

    }

}

**NoSecurityConfig.java**

package com.cognizant.springlearn.Noconfig;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

@Configuration

public class NoSecurityConfig {

    @Bean

    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

        http

            .csrf(csrf -> csrf.disable())

            .headers(headers -> headers.frameOptions().disable())

            .authorizeHttpRequests(auth -> auth.anyRequest().permitAll());

        return http.build();

    }

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=create

server.port=9095

**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

                             http://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>3.2.5</version>

    <relativePath/>

  </parent>

  <groupId>com.cognizant</groupId>

  <artifactId>springlearn-country-lookup</artifactId>

  <version>1.0</version>

  <name>springlearn-country-lookup</name>

  <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-starter-data-jpa</artifactId>

    </dependency>

    <dependency>

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

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

    </dependency>

    <dependency>

      <groupId>com.h2database</groupId>

      <artifactId>h2</artifactId>

      <scope>runtime</scope>

    </dependency>

    <dependency>

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

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

    </dependency>

    <dependency>

      <groupId>jakarta.persistence</groupId>

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

      <version>3.1.0</version>

    </dependency>

    <dependency>

      <groupId>org.junit.jupiter</groupId>

      <artifactId>junit-jupiter</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>3.10.1</version>

        <configuration>

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

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

        </configuration>

      </plugin>

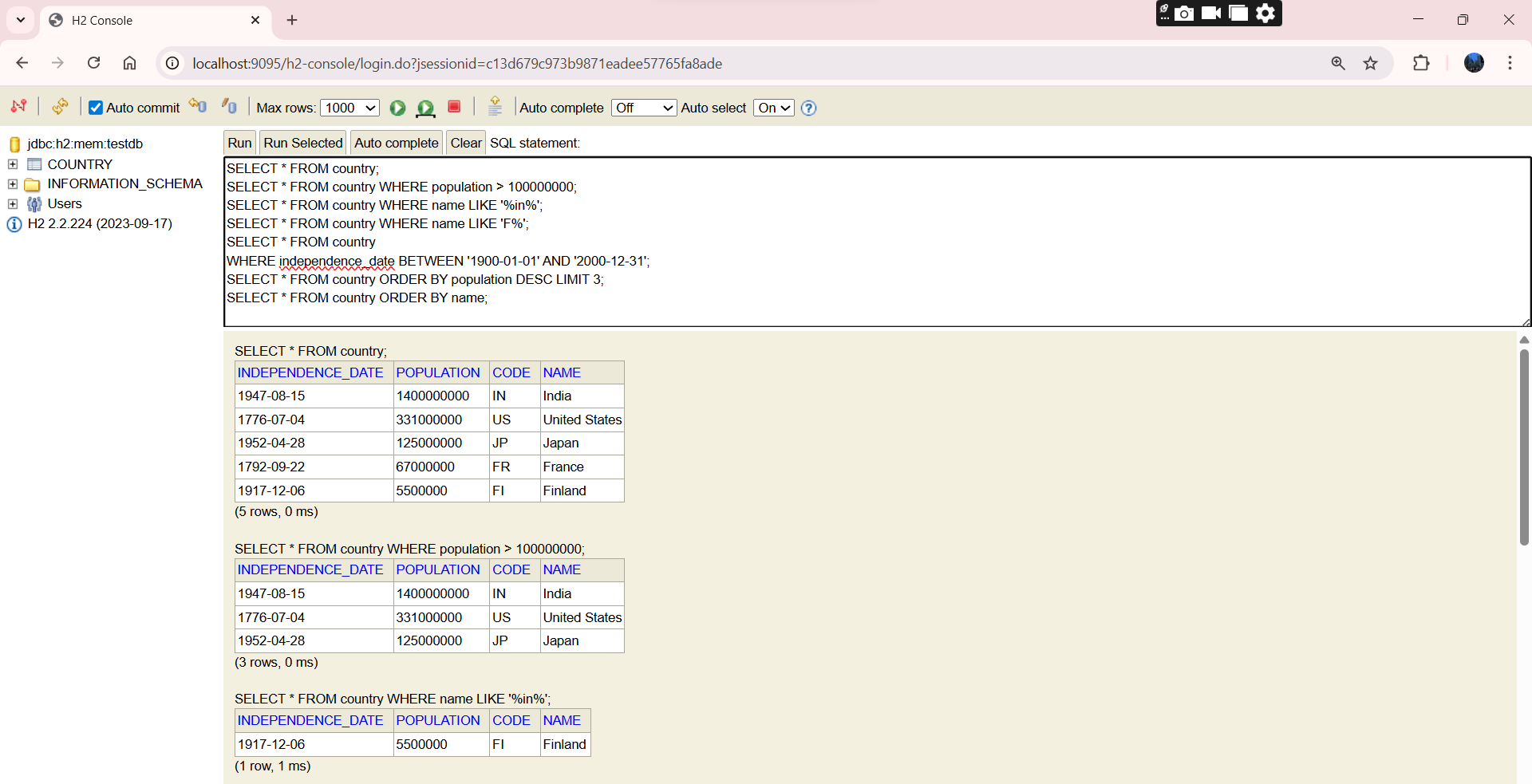
    </plugins>

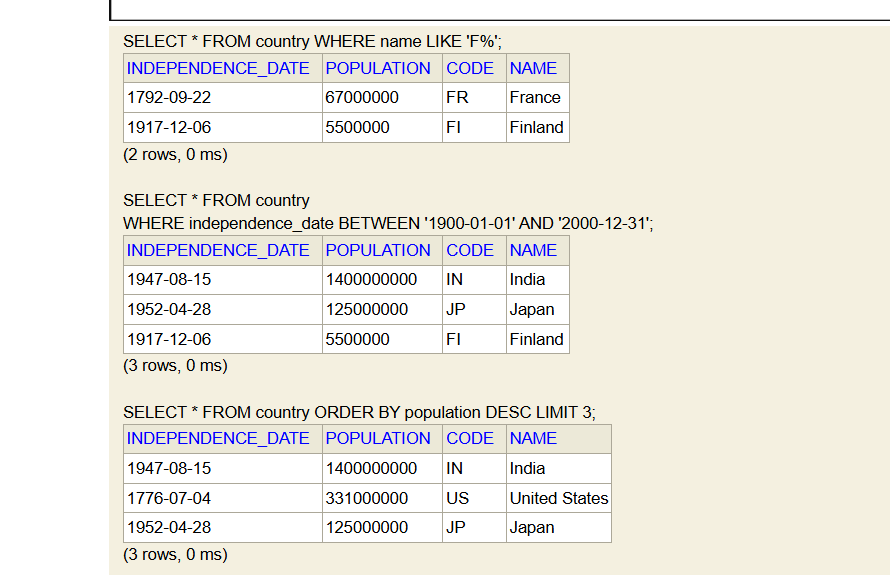
  </build>

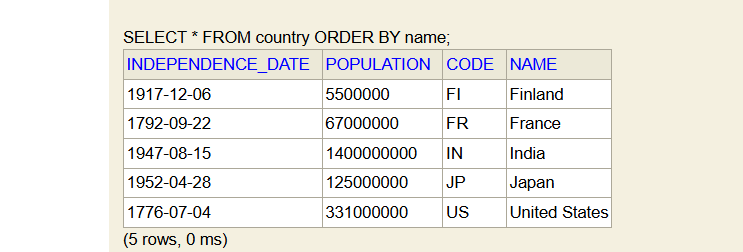
</project>

**OUTPUT:**









* **Demonstrate implementation of O/R Mapping**

**SpringLearnOrmApplication.java**

package com.cognizant.springlearn;

import com.cognizant.springlearn.model.\*;

import com.cognizant.springlearn.repository.\*;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.transaction.annotation.Transactional;

import java.util.Arrays;

@SpringBootApplication

public class SpringlearnOrmApplication implements CommandLineRunner {

    @Autowired

    private EmployeeRepository employeeRepository;

    @Autowired

    private DepartmentRepository departmentRepository;

    @Autowired

    private SkillRepository skillRepository;

    public static void main(String[] args) {

        SpringApplication.run(SpringlearnOrmApplication.class, args);

    }

    @Override

    @Transactional

    public void run(String... args) {

        // Create and save Department

        Department dept = new Department();

        dept.setName("IT");

        departmentRepository.save(dept);

        // Create and save Skills

        Skill skill1 = new Skill();

        skill1.setName("Java");

        Skill skill2 = new Skill();

        skill2.setName("Spring Boot");

        skillRepository.saveAll(Arrays.asList(skill1, skill2));

        // Create and save Employee with department and skills

        Employee emp = new Employee();

        emp.setName("Avni");

        emp.setDepartment(dept);

        emp.setSkills(Arrays.asList(skill1, skill2));

        employeeRepository.save(emp);

        System.out.println("Employee saved with Department and Skills");

    }

}

**Department.java**

package com.cognizant.springlearn.model;

import jakarta.persistence.\*;

import java.util.List;

@Entity

public class Department {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;

    private String name;

    @OneToMany(mappedBy = "department", fetch = FetchType.LAZY)

    private List<Employee> employees;

    public Department() {}

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

}

**Employee.java**

package com.cognizant.springlearn.model;

import jakarta.persistence.\*;

import java.util.List;

@Entity

public class Employee {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;

    private String name;

    @ManyToOne

    @JoinColumn(name = "dept\_id")

    private Department department;

    @ManyToMany

    @JoinTable(

        name = "employee\_skill",

        joinColumns = @JoinColumn(name = "emp\_id"),

        inverseJoinColumns = @JoinColumn(name = "skill\_id")

    )

    private List<Skill> skills;

    public Employee() {}

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public void setDepartment(Department department) {

        this.department = department;

    }

    public void setSkills(List<Skill> skills) {

        this.skills = skills;

    }

}

**Skill.java**

package com.cognizant.springlearn.model;

import jakarta.persistence.\*;

@Entity

public class Skill {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;

    private String name;

    public Skill() {}

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

}

**DepartmentRepository.java**

package com.cognizant.springlearn.repository;

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

import com.cognizant.springlearn.model.Department;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {}

**EmployeeRepository.java**

package com.cognizant.springlearn.repository;

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

import com.cognizant.springlearn.model.Employee;

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

**SkillRepository.java**

package com.cognizant.springlearn.repository;

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

import com.cognizant.springlearn.model.Skill;

public interface SkillRepository extends JpaRepository<Skill, Integer> {}

**EmployeeService.java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Department;

import com.cognizant.springlearn.model.Employee;

import com.cognizant.springlearn.model.Skill;

import com.cognizant.springlearn.repository.DepartmentRepository;

import com.cognizant.springlearn.repository.EmployeeRepository;

import com.cognizant.springlearn.repository.SkillRepository;

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

import org.springframework.stereotype.Service;

import java.util.Arrays;

@Service

public class EmployeeService {

    @Autowired

    private EmployeeRepository employeeRepository;

    @Autowired

    private DepartmentRepository departmentRepository;

    @Autowired

    private SkillRepository skillRepository;

    public void insertSampleEmployee() {

        Department dept = new Department();

        dept.setName("IT");

        departmentRepository.save(dept);

        Skill skill1 = new Skill();

        skill1.setName("Java");

        Skill skill2 = new Skill();

        skill2.setName("SQL");

        skillRepository.saveAll(Arrays.asList(skill1, skill2));

        Employee emp = new Employee();

        emp.setName("John Doe");

        emp.setDepartment(dept);

        emp.setSkills(Arrays.asList(skill1, skill2));

        employeeRepository.save(emp);

        System.out.println("Employee saved with Department and Skills");

    }

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=create

server.port=9095

**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

                             http://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>3.2.5</version>

    <relativePath/>

  </parent>

  <groupId>com.cognizant</groupId>

  <artifactId>springlearn-orm</artifactId>

  <version>1.0</version>

  <name>springlearn-orm</name>

  <properties>

    <java.version>17</java.version>

  </properties>

  <dependencies>

    <!-- Core Spring Boot -->

    <dependency>

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

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

    </dependency>

    <!-- Spring Data JPA -->

    <dependency>

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

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

    </dependency>

    <!-- Web (for REST API / H2 Console) -->

    <dependency>

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

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

    </dependency>

    <!-- In-Memory Database -->

    <dependency>

      <groupId>com.h2database</groupId>

      <artifactId>h2</artifactId>

      <scope>runtime</scope>

    </dependency>

    <!-- Testing -->

    <dependency>

      <groupId>org.junit.jupiter</groupId>

      <artifactId>junit-jupiter</artifactId>

      <scope>test</scope>

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <!-- Spring Boot Plugin -->

      <plugin>

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

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

      </plugin>

      <!-- Java Compiler Plugin -->

      <plugin>

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

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

        <version>3.10.1</version>

        <configuration>

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

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

        </configuration>

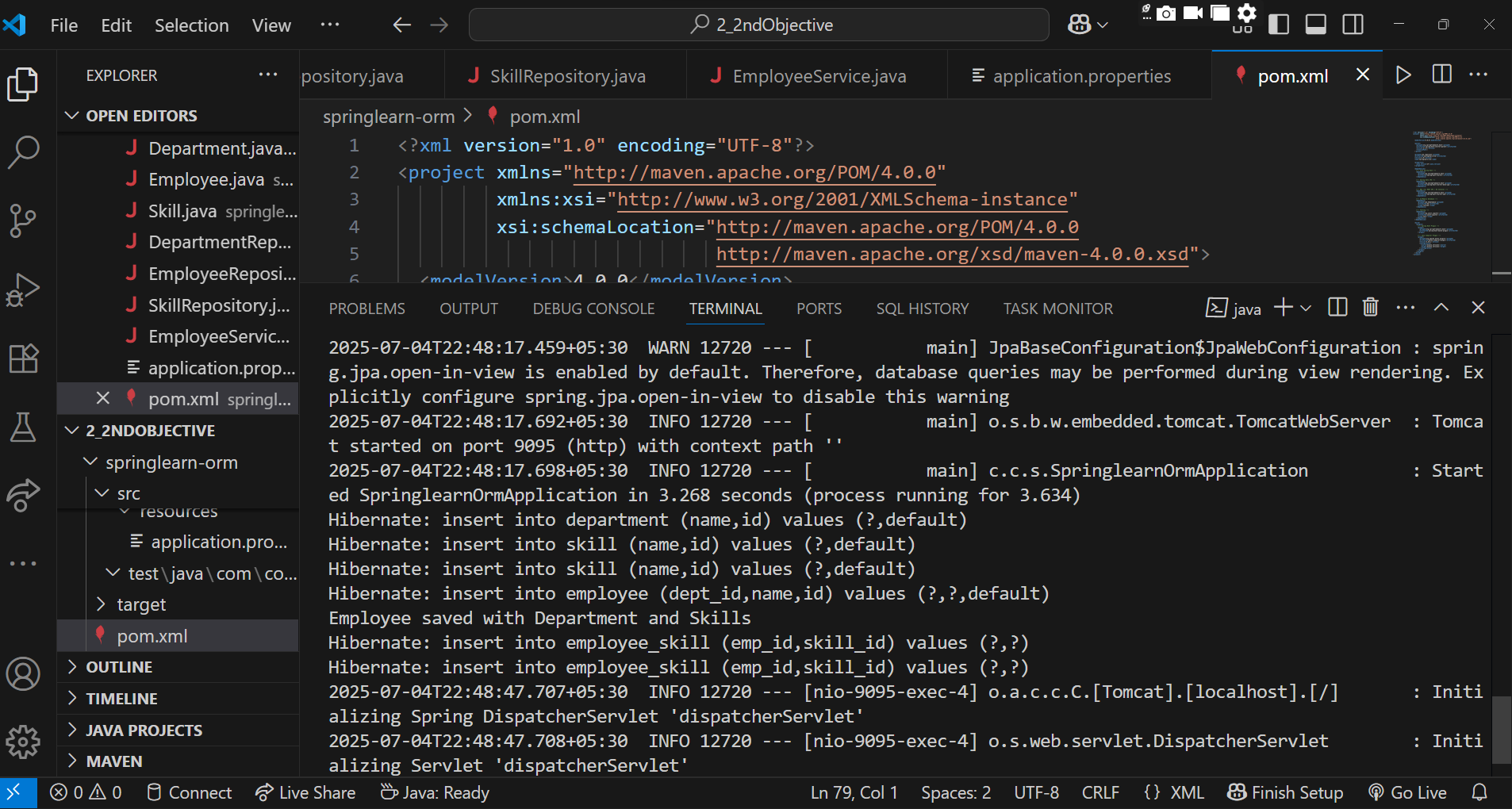
      </plugin>

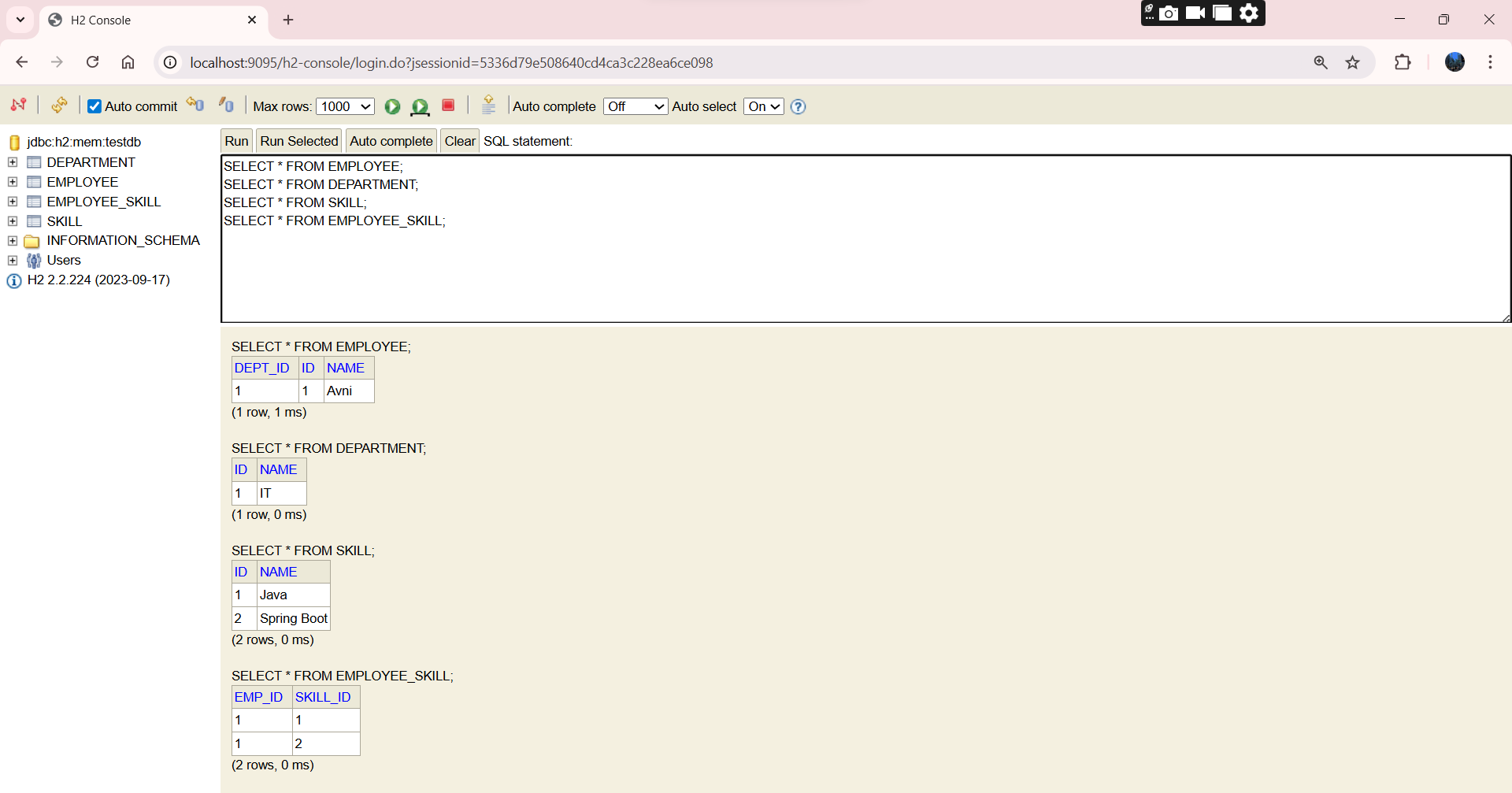
    </plugins>

  </build>

</project>

**OUTPUT:**





**3). 3.Spring-Data-jpa-Handson**

**Objectives**

* **Demonstrate writing Hibernate Query Language and Native Query**

**SpringLearnApplication.java**

package com.cognizant.springlearn;

import com.cognizant.springlearn.model.Employee;

import com.cognizant.springlearn.repository.EmployeeRepository;

import com.cognizant.springlearn.service.EmployeeService;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.math.BigDecimal;

import java.util.List;

@SpringBootApplication

public class SpringlearnApplication implements CommandLineRunner {

    @Autowired

    private EmployeeRepository employeeRepository;

    @Autowired

    private EmployeeService employeeService;

    public static void main(String[] args) {

        SpringApplication.run(SpringlearnApplication.class, args);

    }

    @Override

    public void run(String... args) {

        List<Employee> employees = List.of(

            createEmp("Ayrav", "IT", 60000),

            createEmp("Aryan", "HR", 45000),

            createEmp("Vanya", "IT", 70000)

        );

        employeeRepository.saveAll(employees);

        employeeService.demoQueries();

    }

    private Employee createEmp(String name, String dept, int salary) {

        Employee e = new Employee();

        e.setName(name);

        e.setDepartment(dept);

        e.setSalary(BigDecimal.valueOf(salary));

        return e;

    }

}

**Employee.java**

package com.cognizant.springlearn.model;

import jakarta.persistence.\*;

import java.math.BigDecimal;

@Entity

@Table(name = "employee")

public class Employee {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    private String department;

    private BigDecimal salary;

    public Long getId() { return id; }

    public void setId(Long id) { this.id = id; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getDepartment() { return department; }

    public void setDepartment(String department) { this.department = department; }

    public BigDecimal getSalary() { return salary; }

    public void setSalary(BigDecimal salary) { this.salary = salary; }

}

**EmployeeRepository.java**

package com.cognizant.springlearn.repository;

import com.cognizant.springlearn.model.Employee;

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

import org.springframework.data.jpa.repository.Query;

import java.math.BigDecimal;

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

    @Query("SELECT e FROM Employee e WHERE e.salary > :salary")

    List<Employee> findEmployeesWithSalaryGreaterThan(BigDecimal salary);

    @Query("SELECT COUNT(e) FROM Employee e WHERE e.department = :dept")

    Long countByDepartment(String dept);

    @Query("SELECT AVG(e.salary) FROM Employee e")

    Double findAverageSalary();

    @Query(value = "SELECT \* FROM employee WHERE department = ?1", nativeQuery = true)

    List<Employee> findByDepartmentNative(String department);

}

**EmployeeService.java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Employee;

import com.cognizant.springlearn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import java.math.BigDecimal;

import java.util.List;

@Service

public class EmployeeService {

    @Autowired

    private EmployeeRepository repository;

    public void demoQueries() {

        System.out.println("Employees with salary > 50000:");

        repository.findEmployeesWithSalaryGreaterThan(BigDecimal.valueOf(50000))

                  .forEach(e -> System.out.println(e.getName() + " - " + e.getSalary()));

        System.out.println("Total in IT dept: " + repository.countByDepartment("IT"));

        System.out.println("Average salary: " + repository.findAverageSalary());

        System.out.println("Native query (IT dept):");

        repository.findByDepartmentNative("IT")

                  .forEach(e -> System.out.println(e.getName()));

    }

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=create

server.port=9095

**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

                             http://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>3.2.5</version>

    <relativePath/>

  </parent>

  <groupId>com.cognizant</groupId>

  <artifactId>springlearn-orm</artifactId>

  <version>1.0</version>

  <name>springlearn-orm</name>

  <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-starter-data-jpa</artifactId>

    </dependency>

    <dependency>

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

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

    </dependency>

    <dependency>

      <groupId>com.h2database</groupId>

      <artifactId>h2</artifactId>

      <scope>runtime</scope>

    </dependency>

    <dependency>

      <groupId>org.junit.jupiter</groupId>

      <artifactId>junit-jupiter</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>3.10.1</version>

        <configuration>

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

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

        </configuration>

      </plugin>

    </plugins>

  </build>

</project>

**OUTPUT:**

