**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.

Sol.

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>org.example</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.32</version>  
 </dependency>  
 </dependencies>  
  
 <properties>  
 <maven.compiler.source>24</maven.compiler.source>  
 <maven.compiler.target>24</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
</project>

ApplicationContextXML.java

<?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.library.repository.BookRepository" />  
  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

BookRepository.java

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook(String title) {  
 System.*out*.println("Book saved: " + title);  
 }  
}

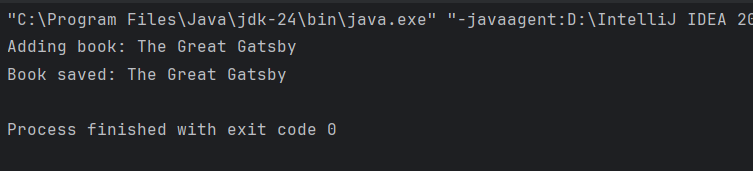
BookService.java

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String title) {  
 System.*out*.println("Adding book: " + title);  
 bookRepository.saveBook(title);  
 }  
}

Main.java

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class MainApp {  
 public static void main(String[] args) {  
 ApplicationContext context =  
 new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService = context.getBean("bookService", BookService.class);  
 bookService.addBook("The Great Gatsby");  
 }  
}

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.

Sol.

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>org.example</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.32</version>  
 </dependency>  
 </dependencies>  
  
 <properties>  
 <maven.compiler.source>24</maven.compiler.source>  
 <maven.compiler.target>24</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
</project>

ApplicationContextXML.java

<?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.library.repository.BookRepository" />  
  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

BookRepository.java

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook(String title) {  
 System.*out*.println("Book saved: " + title);  
 }  
}

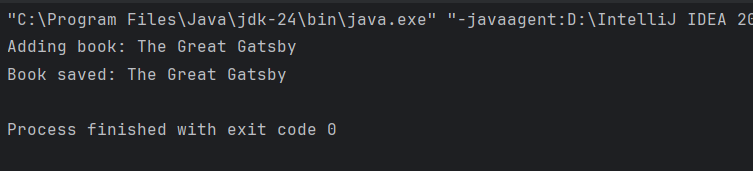
BookService.java

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String title) {  
 System.*out*.println("Adding book: " + title);  
 bookRepository.saveBook(title);  
 }  
}

Main.java

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class MainApp {  
 public static void main(String[] args) {  
 ApplicationContext context =  
 new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService = context.getBean("bookService", BookService.class);  
 bookService.addBook("The Great Gatsby");  
 }  
}

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:**

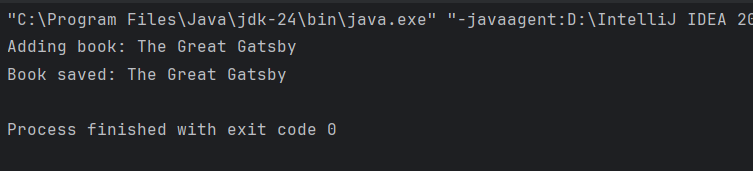
1. **Create a New Maven Project:**
   * Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
   * Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
   * Configure the Maven Compiler Plugin for Java version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

**Sol.**

**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.library</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <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>  
 </plugins>  
 </build>  
</project>**

**Output:**



**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.

Sol.

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>org.example</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.32</version>  
 </dependency>  
 </dependencies>  
  
 <properties>  
 <maven.compiler.source>24</maven.compiler.source>  
 <maven.compiler.target>24</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
</project>

ApplicationContextXML.java

<?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.library.repository.BookRepository" />  
  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

BookRepository.java

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook(String title) {  
 System.*out*.println("Book saved: " + title);  
 }  
}

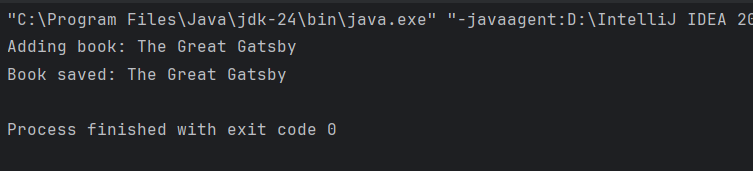
BookService.java

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String title) {  
 System.*out*.println("Adding book: " + title);  
 bookRepository.saveBook(title);  
 }  
}

Main.java

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class MainApp {  
 public static void main(String[] args) {  
 ApplicationContext context =  
 new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService = context.getBean("bookService", BookService.class);  
 bookService.addBook("The Great Gatsby");  
 }  
}

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.

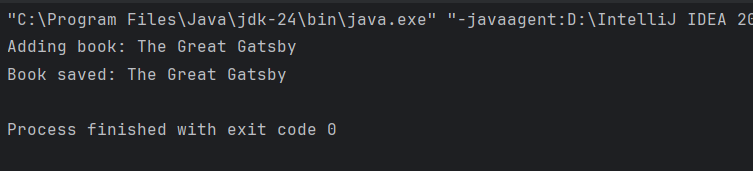
Sol.

BookService.java

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public BookService(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
}  
  
 public void addBook(String title) {  
 System.*out*.println("Adding book: " + title);  
 bookRepository.saveBook(title);  
 }  
}

ApplicationContextXML.java

<?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.library.repository.BookRepository" />  
  
 <bean id="bookService" class="com.library.service.BookService">  
 <contructor-arg ref="bookRepository" />  
 </bean>  
  
</beans>



**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.

Sol.

Pom.xml

<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>

Application.properties

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.hibernate.ddl-auto=update  
  
spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console

Book.java

package com.library.entity;  
  
import jakarta.persistence.\*;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
  
  
@Entity  
public class Book {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String title;  
 private String author;  
  
 public Book() {}  
  
 public Book(String title, String author) {  
 this.title = title;  
 this.author = author;  
 }  
  
 public Long getId() { return 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; }  
}

BookRepository.java

package com.library.repository;  
  
import com.library.entity.Book;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface BookRepository extends JpaRepository<Book, Long> {}

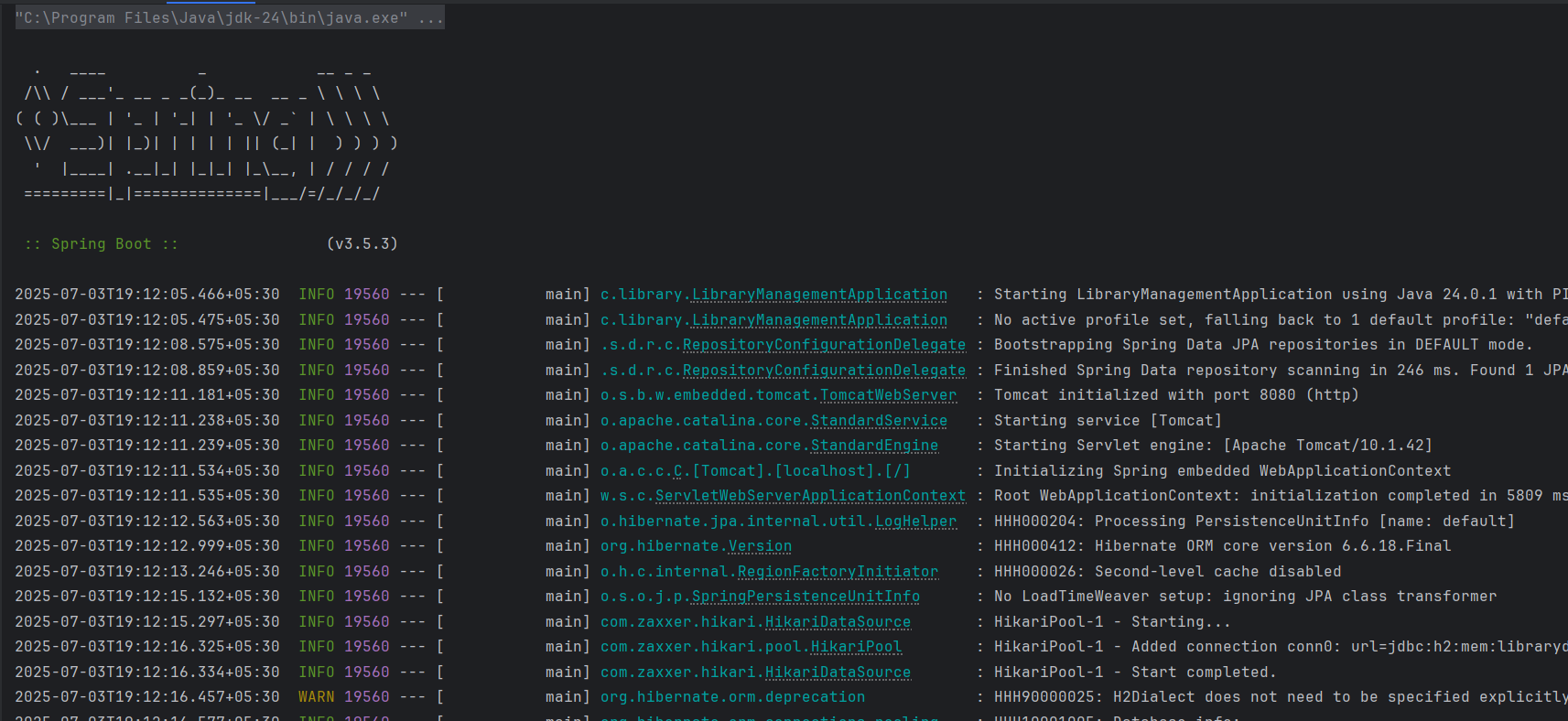
BookController.java

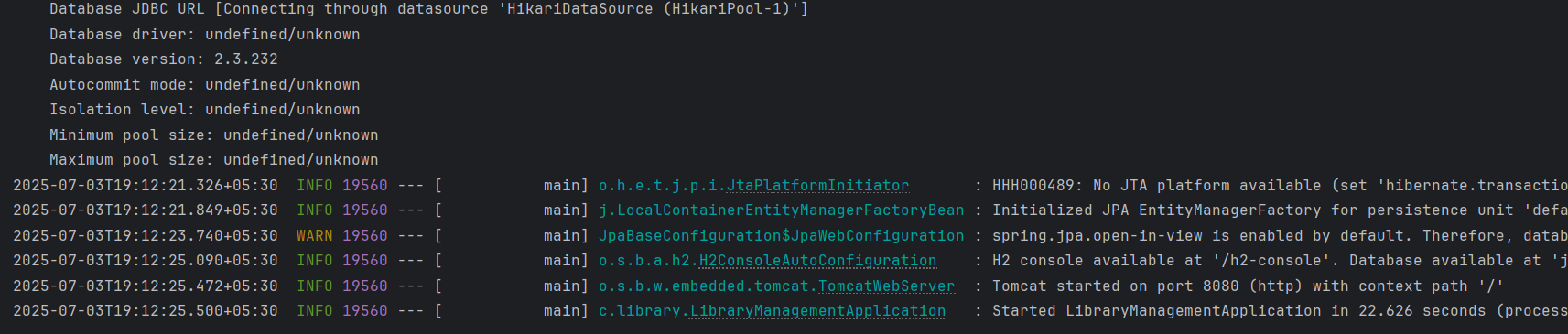
package com.library.controller;  
  
import com.library.entity.Book;  
import com.library.repository.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/books")  
public class BookController {  
  
 @Autowired  
 private BookRepository bookRepository;  
  
 @GetMapping  
 public List<Book> getAllBooks() {  
 return bookRepository.findAll();  
 }  
  
 @GetMapping("/{id}")  
 public Book getBook(@PathVariable Long id) {  
 return bookRepository.findById(id).orElse(null);  
 }  
  
 @PostMapping  
 public Book createBook(@RequestBody Book book) {  
 return bookRepository.save(book);  
 }  
  
 @PutMapping("/{id}")  
 public Book updateBook(@PathVariable Long id, @RequestBody Book book) {  
 Book existing = bookRepository.findById(id).orElse(null);  
 if (existing != null) {  
 existing.setTitle(book.getTitle());  
 existing.setAuthor(book.getAuthor());  
 return bookRepository.save(existing);  
 }  
 return null;  
 }  
  
 @DeleteMapping("/{id}")  
 public void deleteBook(@PathVariable Long id) {  
 bookRepository.deleteById(id);  
 }  
}

LIbraryManagement.java

package com.library;  
  
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);  
 }  
}

Output:





**Hands on 5**

**Implement services for managing Country**   
  
An application requires for features to be implemented with regards to country. These features needs to be supported by implementing them as service using Spring Data JPA.

* Find a country based on country code
* Add new country
* Update country
* Delete country
* Find list of countries matching a partial country name

Before starting the implementation of the above features, there are few configuration and data population that needs to be incorporated. Please refer each topic below and implement the same.   
  
**Explanation for Hibernate table creation configuration**

* Moreover the ddl-auto defines how hibernate behaves if a specific table or column is not present in the database.
  + create - drops existing tables data and structure, then creates new tables
  + validate - check if the table and columns exist or not, throws an exception if a matching table or column is not found
  + update - if a table does not exists, it creates a new table; if a column does not exists, it creates a new column
  + create-drop - creates the table, once all operations are completed, the table is dropped

# Hibernate ddl auto (create, create-drop, update, validate)

spring.jpa.hibernate.ddl-auto=validate

Populate country table

* Delete all the records in Country table and then use the below script to create the actual list of all countries in our world.

insert into country (co\_code, co\_name) values ("AF", "Afghanistan");

insert into country (co\_code, co\_name) values ("AL", "Albania");

insert into country (co\_code, co\_name) values ("DZ", "Algeria");

insert into country (co\_code, co\_name) values ("AS", "American Samoa");

insert into country (co\_code, co\_name) values ("AD", "Andorra");

insert into country (co\_code, co\_name) values ("AO", "Angola");

insert into country (co\_code, co\_name) values ("AI", "Anguilla");

insert into country (co\_code, co\_name) values ("AQ", "Antarctica");

insert into country (co\_code, co\_name) values ("AG", "Antigua and Barbuda");

insert into country (co\_code, co\_name) values ("AR", "Argentina");

insert into country (co\_code, co\_name) values ("AM", "Armenia");

insert into country (co\_code, co\_name) values ("AW", "Aruba");

insert into country (co\_code, co\_name) values ("AU", "Australia");

insert into country (co\_code, co\_name) values ("AT", "Austria");

insert into country (co\_code, co\_name) values ("AZ", "Azerbaijan");

insert into country (co\_code, co\_name) values ("BS", "Bahamas");

insert into country (co\_code, co\_name) values ("BH", "Bahrain");

insert into country (co\_code, co\_name) values ("BD", "Bangladesh");

insert into country (co\_code, co\_name) values ("BB", "Barbados");

insert into country (co\_code, co\_name) values ("BY", "Belarus");

insert into country (co\_code, co\_name) values ("BE", "Belgium");

insert into country (co\_code, co\_name) values ("BZ", "Belize");

insert into country (co\_code, co\_name) values ("BJ", "Benin");

insert into country (co\_code, co\_name) values ("BM", "Bermuda");

insert into country (co\_code, co\_name) values ("BT", "Bhutan");

insert into country (co\_code, co\_name) values ("BO", "Bolivia, Plurinational State of");

insert into country (co\_code, co\_name) values ("BQ", "Bonaire, Sint Eustatius and Saba");

insert into country (co\_code, co\_name) values ("BA", "Bosnia and Herzegovina");

insert into country (co\_code, co\_name) values ("BW", "Botswana");

insert into country (co\_code, co\_name) values ("BV", "Bouvet Island");

insert into country (co\_code, co\_name) values ("BR", "Brazil");

insert into country (co\_code, co\_name) values ("IO", "British Indian Ocean Territory");

insert into country (co\_code, co\_name) values ("BN", "Brunei Darussalam");

insert into country (co\_code, co\_name) values ("BG", "Bulgaria");

insert into country (co\_code, co\_name) values ("BF", "Burkina Faso");

insert into country (co\_code, co\_name) values ("BI", "Burundi");

insert into country (co\_code, co\_name) values ("KH", "Cambodia");

insert into country (co\_code, co\_name) values ("CM", "Cameroon");

insert into country (co\_code, co\_name) values ("CA", "Canada");

insert into country (co\_code, co\_name) values ("CV", "Cape Verde");

insert into country (co\_code, co\_name) values ("KY", "Cayman Islands");

insert into country (co\_code, co\_name) values ("CF", "Central African Republic");

insert into country (co\_code, co\_name) values ("TD", "Chad");

insert into country (co\_code, co\_name) values ("CL", "Chile");

insert into country (co\_code, co\_name) values ("CN", "China");

insert into country (co\_code, co\_name) values ("CX", "Christmas Island");

insert into country (co\_code, co\_name) values ("CC", "Cocos (Keeling) Islands");

insert into country (co\_code, co\_name) values ("CO", "Colombia");

insert into country (co\_code, co\_name) values ("KM", "Comoros");

insert into country (co\_code, co\_name) values ("CG", "Congo");

insert into country (co\_code, co\_name) values ("CD", "Congo, the Democratic Republic of the");

insert into country (co\_code, co\_name) values ("CK", "Cook Islands");

insert into country (co\_code, co\_name) values ("CR", "Costa Rica");

insert into country (co\_code, co\_name) values ("HR", "Croatia");

insert into country (co\_code, co\_name) values ("CU", "Cuba");

insert into country (co\_code, co\_name) values ("CW", "Curaçao");

insert into country (co\_code, co\_name) values ("CY", "Cyprus");

insert into country (co\_code, co\_name) values ("CZ", "Czech Republic");

insert into country (co\_code, co\_name) values ("CI", "Côte d'Ivoire");

insert into country (co\_code, co\_name) values ("DK", "Denmark");

insert into country (co\_code, co\_name) values ("DJ", "Djibouti");

insert into country (co\_code, co\_name) values ("DM", "Dominica");

insert into country (co\_code, co\_name) values ("DO", "Dominican Republic");

insert into country (co\_code, co\_name) values ("EC", "Ecuador");

insert into country (co\_code, co\_name) values ("EG", "Egypt");

insert into country (co\_code, co\_name) values ("SV", "El Salvador");

insert into country (co\_code, co\_name) values ("GQ", "Equatorial Guinea");

insert into country (co\_code, co\_name) values ("ER", "Eritrea");

insert into country (co\_code, co\_name) values ("EE", "Estonia");

insert into country (co\_code, co\_name) values ("ET", "Ethiopia");

insert into country (co\_code, co\_name) values ("FK", "Falkland Islands (Malvinas)");

insert into country (co\_code, co\_name) values ("FO", "Faroe Islands");

insert into country (co\_code, co\_name) values ("FJ", "Fiji");

insert into country (co\_code, co\_name) values ("FI", "Finland");

insert into country (co\_code, co\_name) values ("FR", "France");

insert into country (co\_code, co\_name) values ("GF", "French Guiana");

insert into country (co\_code, co\_name) values ("PF", "French Polynesia");

insert into country (co\_code, co\_name) values ("TF", "French Southern Territories");

insert into country (co\_code, co\_name) values ("GA", "Gabon");

insert into country (co\_code, co\_name) values ("GM", "Gambia");

insert into country (co\_code, co\_name) values ("GE", "Georgia");

insert into country (co\_code, co\_name) values ("DE", "Germany");

insert into country (co\_code, co\_name) values ("GH", "Ghana");

insert into country (co\_code, co\_name) values ("GI", "Gibraltar");

insert into country (co\_code, co\_name) values ("GR", "Greece");

insert into country (co\_code, co\_name) values ("GL", "Greenland");

insert into country (co\_code, co\_name) values ("GD", "Grenada");

insert into country (co\_code, co\_name) values ("GP", "Guadeloupe");

insert into country (co\_code, co\_name) values ("GU", "Guam");

insert into country (co\_code, co\_name) values ("GT", "Guatemala");

insert into country (co\_code, co\_name) values ("GG", "Guernsey");

insert into country (co\_code, co\_name) values ("GN", "Guinea");

insert into country (co\_code, co\_name) values ("GW", "Guinea-Bissau");

insert into country (co\_code, co\_name) values ("GY", "Guyana");

insert into country (co\_code, co\_name) values ("HT", "Haiti");

insert into country (co\_code, co\_name) values ("HM", "Heard Island and McDonald Islands");

insert into country (co\_code, co\_name) values ("VA", "Holy See (Vatican City State)");

insert into country (co\_code, co\_name) values ("HN", "Honduras");

insert into country (co\_code, co\_name) values ("HK", "Hong Kong");

insert into country (co\_code, co\_name) values ("HU", "Hungary");

insert into country (co\_code, co\_name) values ("IS", "Iceland");

insert into country (co\_code, co\_name) values ("IN", "India");

insert into country (co\_code, co\_name) values ("ID", "Indonesia");

insert into country (co\_code, co\_name) values ("IR", "Iran, Islamic Republic of");

insert into country (co\_code, co\_name) values ("IQ", "Iraq");

insert into country (co\_code, co\_name) values ("IE", "Ireland");

insert into country (co\_code, co\_name) values ("IM", "Isle of Man");

insert into country (co\_code, co\_name) values ("IL", "Israel");

insert into country (co\_code, co\_name) values ("IT", "Italy");

insert into country (co\_code, co\_name) values ("JM", "Jamaica");

insert into country (co\_code, co\_name) values ("JP", "Japan");

insert into country (co\_code, co\_name) values ("JE", "Jersey");

insert into country (co\_code, co\_name) values ("JO", "Jordan");

insert into country (co\_code, co\_name) values ("KZ", "Kazakhstan");

insert into country (co\_code, co\_name) values ("KE", "Kenya");

insert into country (co\_code, co\_name) values ("KI", "Kiribati");

insert into country (co\_code, co\_name) values ("KP", "Democratic People's Republic of Korea");

insert into country (co\_code, co\_name) values ("KR", "Republic of Korea");

insert into country (co\_code, co\_name) values ("KW", "Kuwait");

insert into country (co\_code, co\_name) values ("KG", "Kyrgyzstan");

insert into country (co\_code, co\_name) values ("LA", "Lao People's Democratic Republic");

insert into country (co\_code, co\_name) values ("LV", "Latvia");

insert into country (co\_code, co\_name) values ("LB", "Lebanon");

insert into country (co\_code, co\_name) values ("LS", "Lesotho");

insert into country (co\_code, co\_name) values ("LR", "Liberia");

insert into country (co\_code, co\_name) values ("LY", "Libya");

insert into country (co\_code, co\_name) values ("LI", "Liechtenstein");

insert into country (co\_code, co\_name) values ("LT", "Lithuania");

insert into country (co\_code, co\_name) values ("LU", "Luxembourg");

insert into country (co\_code, co\_name) values ("MO", "Macao");

insert into country (co\_code, co\_name) values ("MK", "Macedonia, the Former Yugoslav Republic of");

insert into country (co\_code, co\_name) values ("MG", "Madagascar");

insert into country (co\_code, co\_name) values ("MW", "Malawi");

insert into country (co\_code, co\_name) values ("MY", "Malaysia");

insert into country (co\_code, co\_name) values ("MV", "Maldives");

insert into country (co\_code, co\_name) values ("ML", "Mali");

insert into country (co\_code, co\_name) values ("MT", "Malta");

insert into country (co\_code, co\_name) values ("MH", "Marshall Islands");

insert into country (co\_code, co\_name) values ("MQ", "Martinique");

insert into country (co\_code, co\_name) values ("MR", "Mauritania");

insert into country (co\_code, co\_name) values ("MU", "Mauritius");

insert into country (co\_code, co\_name) values ("YT", "Mayotte");

insert into country (co\_code, co\_name) values ("MX", "Mexico");

insert into country (co\_code, co\_name) values ("FM", "Micronesia, Federated States of");

insert into country (co\_code, co\_name) values ("MD", "Moldova, Republic of");

insert into country (co\_code, co\_name) values ("MC", "Monaco");

insert into country (co\_code, co\_name) values ("MN", "Mongolia");

insert into country (co\_code, co\_name) values ("ME", "Montenegro");

insert into country (co\_code, co\_name) values ("MS", "Montserrat");

insert into country (co\_code, co\_name) values ("MA", "Morocco");

insert into country (co\_code, co\_name) values ("MZ", "Mozambique");

insert into country (co\_code, co\_name) values ("MM", "Myanmar");

insert into country (co\_code, co\_name) values ("NA", "Namibia");

insert into country (co\_code, co\_name) values ("NR", "Nauru");

insert into country (co\_code, co\_name) values ("NP", "Nepal");

insert into country (co\_code, co\_name) values ("NL", "Netherlands");

insert into country (co\_code, co\_name) values ("NC", "New Caledonia");

insert into country (co\_code, co\_name) values ("NZ", "New Zealand");

insert into country (co\_code, co\_name) values ("NI", "Nicaragua");

insert into country (co\_code, co\_name) values ("NE", "Niger");

insert into country (co\_code, co\_name) values ("NG", "Nigeria");

insert into country (co\_code, co\_name) values ("NU", "Niue");

insert into country (co\_code, co\_name) values ("NF", "Norfolk Island");

insert into country (co\_code, co\_name) values ("MP", "Northern Mariana Islands");

insert into country (co\_code, co\_name) values ("NO", "Norway");

insert into country (co\_code, co\_name) values ("OM", "Oman");

insert into country (co\_code, co\_name) values ("PK", "Pakistan");

insert into country (co\_code, co\_name) values ("PW", "Palau");

insert into country (co\_code, co\_name) values ("PS", "Palestine, State of");

insert into country (co\_code, co\_name) values ("PA", "Panama");

insert into country (co\_code, co\_name) values ("PG", "Papua New Guinea");

insert into country (co\_code, co\_name) values ("PY", "Paraguay");

insert into country (co\_code, co\_name) values ("PE", "Peru");

insert into country (co\_code, co\_name) values ("PH", "Philippines");

insert into country (co\_code, co\_name) values ("PN", "Pitcairn");

insert into country (co\_code, co\_name) values ("PL", "Poland");

insert into country (co\_code, co\_name) values ("PT", "Portugal");

insert into country (co\_code, co\_name) values ("PR", "Puerto Rico");

insert into country (co\_code, co\_name) values ("QA", "Qatar");

insert into country (co\_code, co\_name) values ("RO", "Romania");

insert into country (co\_code, co\_name) values ("RU", "Russian Federation");

insert into country (co\_code, co\_name) values ("RW", "Rwanda");

insert into country (co\_code, co\_name) values ("RE", "Réunion");

insert into country (co\_code, co\_name) values ("BL", "Saint Barthélemy");

insert into country (co\_code, co\_name) values ("SH", "Saint Helena, Ascension and Tristan da Cunha");

insert into country (co\_code, co\_name) values ("KN", "Saint Kitts and Nevis");

insert into country (co\_code, co\_name) values ("LC", "Saint Lucia");

insert into country (co\_code, co\_name) values ("MF", "Saint Martin (French part)");

insert into country (co\_code, co\_name) values ("PM", "Saint Pierre and Miquelon");

insert into country (co\_code, co\_name) values ("VC", "Saint Vincent and the Grenadines");

insert into country (co\_code, co\_name) values ("WS", "Samoa");

insert into country (co\_code, co\_name) values ("SM", "San Marino");

insert into country (co\_code, co\_name) values ("ST", "Sao Tome and Principe");

insert into country (co\_code, co\_name) values ("SA", "Saudi Arabia");

insert into country (co\_code, co\_name) values ("SN", "Senegal");

insert into country (co\_code, co\_name) values ("RS", "Serbia");

insert into country (co\_code, co\_name) values ("SC", "Seychelles");

insert into country (co\_code, co\_name) values ("SL", "Sierra Leone");

insert into country (co\_code, co\_name) values ("SG", "Singapore");

insert into country (co\_code, co\_name) values ("SX", "Sint Maarten (Dutch part)");

insert into country (co\_code, co\_name) values ("SK", "Slovakia");

insert into country (co\_code, co\_name) values ("SI", "Slovenia");

insert into country (co\_code, co\_name) values ("SB", "Solomon Islands");

insert into country (co\_code, co\_name) values ("SO", "Somalia");

insert into country (co\_code, co\_name) values ("ZA", "South Africa");

insert into country (co\_code, co\_name) values ("GS", "South Georgia and the South Sandwich Islands");

insert into country (co\_code, co\_name) values ("SS", "South Sudan");

insert into country (co\_code, co\_name) values ("ES", "Spain");

insert into country (co\_code, co\_name) values ("LK", "Sri Lanka");

insert into country (co\_code, co\_name) values ("SD", "Sudan");

insert into country (co\_code, co\_name) values ("SR", "Suriname");

insert into country (co\_code, co\_name) values ("SJ", "Svalbard and Jan Mayen");

insert into country (co\_code, co\_name) values ("SZ", "Swaziland");

insert into country (co\_code, co\_name) values ("SE", "Sweden");

insert into country (co\_code, co\_name) values ("CH", "Switzerland");

insert into country (co\_code, co\_name) values ("SY", "Syrian Arab Republic");

insert into country (co\_code, co\_name) values ("TW", "Taiwan, Province of China");

insert into country (co\_code, co\_name) values ("TJ", "Tajikistan");

insert into country (co\_code, co\_name) values ("TZ", "Tanzania, United Republic of");

insert into country (co\_code, co\_name) values ("TH", "Thailand");

insert into country (co\_code, co\_name) values ("TL", "Timor-Leste");

insert into country (co\_code, co\_name) values ("TG", "Togo");

insert into country (co\_code, co\_name) values ("TK", "Tokelau");

insert into country (co\_code, co\_name) values ("TO", "Tonga");

insert into country (co\_code, co\_name) values ("TT", "Trinidad and Tobago");

insert into country (co\_code, co\_name) values ("TN", "Tunisia");

insert into country (co\_code, co\_name) values ("TR", "Turkey");

insert into country (co\_code, co\_name) values ("TM", "Turkmenistan");

insert into country (co\_code, co\_name) values ("TC", "Turks and Caicos Islands");

insert into country (co\_code, co\_name) values ("TV", "Tuvalu");

insert into country (co\_code, co\_name) values ("UG", "Uganda");

insert into country (co\_code, co\_name) values ("UA", "Ukraine");

insert into country (co\_code, co\_name) values ("AE", "United Arab Emirates");

insert into country (co\_code, co\_name) values ("GB", "United Kingdom");

insert into country (co\_code, co\_name) values ("US", "United States");

insert into country (co\_code, co\_name) values ("UM", "United States Minor Outlying Islands");

insert into country (co\_code, co\_name) values ("UY", "Uruguay");

insert into country (co\_code, co\_name) values ("UZ", "Uzbekistan");

insert into country (co\_code, co\_name) values ("VU", "Vanuatu");

insert into country (co\_code, co\_name) values ("VE", "Venezuela, Bolivarian Republic of");

insert into country (co\_code, co\_name) values ("VN", "Viet Nam");

insert into country (co\_code, co\_name) values ("VG", "Virgin Islands, British");

insert into country (co\_code, co\_name) values ("VI", "Virgin Islands, U.S.");

insert into country (co\_code, co\_name) values ("WF", "Wallis and Futuna");

insert into country (co\_code, co\_name) values ("EH", "Western Sahara");

insert into country (co\_code, co\_name) values ("YE", "Yemen");

insert into country (co\_code, co\_name) values ("ZM", "Zambia");

insert into country (co\_code, co\_name) values ("ZW", "Zimbabwe");

insert into country (co\_code, co\_name) values ("AX", "Åland Islands");

Refer subsequent hands on exercises to implement the features related to country.

Sol.

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>  
  
   
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.2.5</version>   
 <relativePath/>   
 </parent>  
  
 <groupId>com.cognizant</groupId>  
 <artifactId>spring-learn</artifactId>  
 <version>1.0.0</version>  
 <packaging>jar</packaging>  
 <name>spring-learn</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>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <version>1.18.30</version>   
 <scope>provided</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
</project>

Application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/springlearn  
spring.datasource.username=root1  
spring.datasource.password=0Mg@np@t@yn@m@h@  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.format\_sql=true  
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect  
  
logging.level.org.springframework=INFO  
logging.level.com.cognizant.spring-learn=DEBUG

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 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;  
 }  
}

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 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.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/countries")  
public class CountryController {  
  
 @Autowired  
 private CountryService countryService;  
  
 @GetMapping("/{code}")  
 public ResponseEntity<Country> getCountry(@PathVariable String code) throws CountryNotFoundException {  
 return ResponseEntity.*ok*(countryService.findByCode(code));  
 }  
  
  
 @PostMapping  
 public ResponseEntity<Country> createCountry(@RequestBody Country country) {  
 return ResponseEntity.*ok*(countryService.addCountry(country));  
 }  
  
 @PutMapping  
 public ResponseEntity<Country> updateCountry(@RequestBody Country country) {  
 return ResponseEntity.*ok*(countryService.updateCountry(country));  
 }  
  
 @DeleteMapping("/{code}")  
 public ResponseEntity<Void> deleteCountry(@PathVariable String code) {  
 countryService.deleteCountry(code);  
 return ResponseEntity.*noContent*().build();  
 }  
  
 @GetMapping("/search")  
 public ResponseEntity<List<Country>> searchCountries(@RequestParam String name) {  
 return ResponseEntity.*ok*(countryService.findByPartialName(name));  
 }  
}

CountryService.java

package com.cognizant.springlearn.service;  
  
import com.cognizant.springlearn.model.Country;  
import com.cognizant.springlearn.service.exception.CountryNotFoundException;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface CountryService {  
 Country findByCode(String code) throws CountryNotFoundException;  
 Country addCountry(Country country);  
 Country updateCountry(Country country);  
 void deleteCountry(String code);  
 List<Country> findByPartialName(String name);  
}

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 findByCode(String countryCode) throws CountryNotFoundException {  
 Optional<Country> result = countryRepository.findById(countryCode);  
 if (!result.isPresent()) {  
 throw new CountryNotFoundException("Country with code " + countryCode + " not found");  
 }  
 return result.get();  
 }  
  
  
// @Override  
// public Optional<Country> findByCode(String code) {  
// return countryRepository.findById(code);  
// }  
  
 @Override  
 public Country addCountry(Country country) {  
 return countryRepository.save(country);  
 }  
  
 @Override  
 public Country updateCountry(Country country) {  
 return countryRepository.save(country);  
 }  
  
 @Override  
 public void deleteCountry(String code) {  
 countryRepository.deleteById(code);  
 }  
  
 @Override  
 public List<Country> findByPartialName(String name) {  
 return countryRepository.findByCoNameContainingIgnoreCase(name);  
 }  
}

data.sql

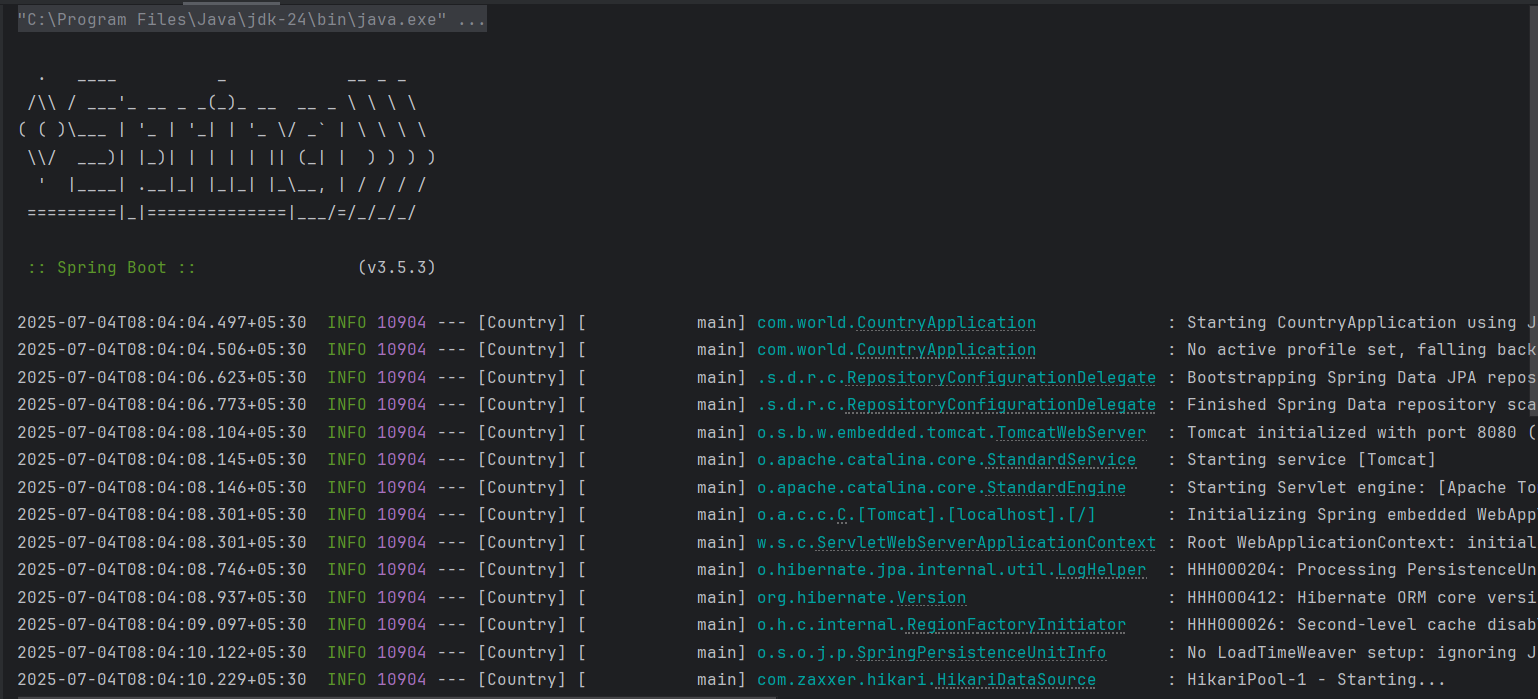
DELETE FROM country;  
  
INSERT INTO country (co\_code, co\_name) VALUES ('AF', 'Afghanistan');  
INSERT INTO country (co\_code, co\_name) VALUES ('AL', 'Albania');  
INSERT INTO country (co\_code, co\_name) VALUES ('DZ', 'Algeria');  
INSERT INTO country (co\_code, co\_name) VALUES ('AS', 'American Samoa');  
INSERT INTO country (co\_code, co\_name) VALUES ('AD', 'Andorra');  
INSERT INTO country (co\_code, co\_name) VALUES ('AO', 'Angola');  
INSERT INTO country (co\_code, co\_name) VALUES ('AI', 'Anguilla');  
INSERT INTO country (co\_code, co\_name) VALUES ('AQ', 'Antarctica');  
INSERT INTO country (co\_code, co\_name) VALUES ('AG', 'Antigua and Barbuda');  
INSERT INTO country (co\_code, co\_name) VALUES ('AR', 'Argentina');  
  
INSERT INTO country (co\_code, co\_name) VALUES ('AE', 'United Arab Emirates');  
INSERT INTO country (co\_code, co\_name) VALUES ('GB', 'United Kingdom');  
INSERT INTO country (co\_code, co\_name) VALUES ('US', 'United States');  
INSERT INTO country (co\_code, co\_name) VALUES ('UM', 'United States Minor Outlying Islands');  
INSERT INTO country (co\_code, co\_name) VALUES ('UY', 'Uruguay');  
INSERT INTO country (co\_code, co\_name) VALUES ('UZ', 'Uzbekistan');  
INSERT INTO country (co\_code, co\_name) VALUES ('VU', 'Vanuatu');  
INSERT INTO country (co\_code, co\_name) VALUES ('VE', 'Venezuela, Bolivarian Republic of');  
INSERT INTO country (co\_code, co\_name) VALUES ('VN', 'Viet Nam');  
INSERT INTO country (co\_code, co\_name) VALUES ('VG', 'Virgin Islands, British');  
INSERT INTO country (co\_code, co\_name) VALUES ('VI', 'Virgin Islands, U.S.');  
INSERT INTO country (co\_code, co\_name) VALUES ('WF', 'Wallis and Futuna');  
INSERT INTO country (co\_code, co\_name) VALUES ('EH', 'Western Sahara');  
INSERT INTO country (co\_code, co\_name) VALUES ('YE', 'Yemen');  
INSERT INTO country (co\_code, co\_name) VALUES ('ZM', 'Zambia');  
INSERT INTO country (co\_code, co\_name) VALUES ('ZW', 'Zimbabwe');

CountryApplication.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.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@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);  
 CountryService countryService = context.getBean(CountryService.class);  
  
 *getAllCountriesTest*(countryService); *testAddCountry*(countryService);

}  
  
  
 private static void getAllCountriesTest(CountryService countryService) {  
 *LOGGER*.info("Start");  
  
 try {  
 Country country = countryService.findByCode("ZW");  
 *LOGGER*.debug("Country:{}", country);  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Exception: {}", e.getMessage());  
 }  
  
 *LOGGER*.info("End");  
 }  
  
 private static void testAddCountry(CountryService countryService) {  
 *LOGGER*.info("Start");  
  
 Country newCountry = new Country();  
 newCountry.setCoCode("JP");  
 newCountry.setCoName("Japan");  
  
 countryService.addCountry(newCountry);  
  
 try {  
 Country country = countryService.findByCode("JP");  
 *LOGGER*.debug("Country added: {}", country);  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Country not found after adding: {}", e.getMessage());  
 }  
  
 *LOGGER*.info("End");  
 }  
  
  
}

Output:



**Hands on 6**

**Find a country based on country code** 

* Create new exception class CountryNotFoundException in com.cognizant.spring-learn.service.exception
* Create new method findCountryByCode() in CountryService with @Transactional annotation
* In findCountryByCode() method, perform the following steps:
  + Method signature

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException

* Get the country based on findById() built in method

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

* From the result, check if a country is found. If not found, throw CountryNotFoundException

if (!result.isPresent())

* Use get() method to return the country fetched.

Country country = result.get();

* Include new test method in OrmLearnApplication to find a country based on country code and compare the country name to check if it is valid.

    private static void getAllCountriesTest() {

        LOGGER.info("Start");

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

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

        LOGGER.info("End");

    }

* Invoke the above method in main() method and test it.

**NOTE:** SME to explain the importance of @Transactional annotation. Spring takes care of creating the Hibernate session and manages the transactionality when executing the service method.

Sol.

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>  
  
   
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.2.5</version>   
 <relativePath/>   
 </parent>  
  
 <groupId>com.cognizant</groupId>  
 <artifactId>spring-learn</artifactId>  
 <version>1.0.0</version>  
 <packaging>jar</packaging>  
 <name>spring-learn</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>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <version>1.18.30</version>   
 <scope>provided</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
</project>

Application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/springlearn  
spring.datasource.username=root1  
spring.datasource.password=0Mg@np@t@yn@m@h@  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.format\_sql=true  
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect  
  
logging.level.org.springframework=INFO  
logging.level.com.cognizant.spring-learn=DEBUG

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 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;  
 }  
}

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 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.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/countries")  
public class CountryController {  
  
 @Autowired  
 private CountryService countryService;  
  
 @GetMapping("/{code}")  
 public ResponseEntity<Country> getCountry(@PathVariable String code) throws CountryNotFoundException {  
 return ResponseEntity.*ok*(countryService.findByCode(code));  
 }  
  
  
 @PostMapping  
 public ResponseEntity<Country> createCountry(@RequestBody Country country) {  
 return ResponseEntity.*ok*(countryService.addCountry(country));  
 }  
  
 @PutMapping  
 public ResponseEntity<Country> updateCountry(@RequestBody Country country) {  
 return ResponseEntity.*ok*(countryService.updateCountry(country));  
 }  
  
 @DeleteMapping("/{code}")  
 public ResponseEntity<Void> deleteCountry(@PathVariable String code) {  
 countryService.deleteCountry(code);  
 return ResponseEntity.*noContent*().build();  
 }  
  
 @GetMapping("/search")  
 public ResponseEntity<List<Country>> searchCountries(@RequestParam String name) {  
 return ResponseEntity.*ok*(countryService.findByPartialName(name));  
 }  
}

CountryService.java

package com.cognizant.springlearn.service;  
  
import com.cognizant.springlearn.model.Country;  
import com.cognizant.springlearn.service.exception.CountryNotFoundException;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface CountryService {  
 Country findByCode(String code) throws CountryNotFoundException;  
 Country addCountry(Country country);  
 Country updateCountry(Country country);  
 void deleteCountry(String code);  
 List<Country> findByPartialName(String name);  
}

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 findByCode(String countryCode) throws CountryNotFoundException {  
 Optional<Country> result = countryRepository.findById(countryCode);  
 if (!result.isPresent()) {  
 throw new CountryNotFoundException("Country with code " + countryCode + " not found");  
 }  
 return result.get();  
 }  
  
  
// @Override  
// public Optional<Country> findByCode(String code) {  
// return countryRepository.findById(code);  
// }  
  
 @Override  
 public Country addCountry(Country country) {  
 return countryRepository.save(country);  
 }  
  
 @Override  
 public Country updateCountry(Country country) {  
 return countryRepository.save(country);  
 }  
  
 @Override  
 public void deleteCountry(String code) {  
 countryRepository.deleteById(code);  
 }  
  
 @Override  
 public List<Country> findByPartialName(String name) {  
 return countryRepository.findByCoNameContainingIgnoreCase(name);  
 }  
}

data.sql

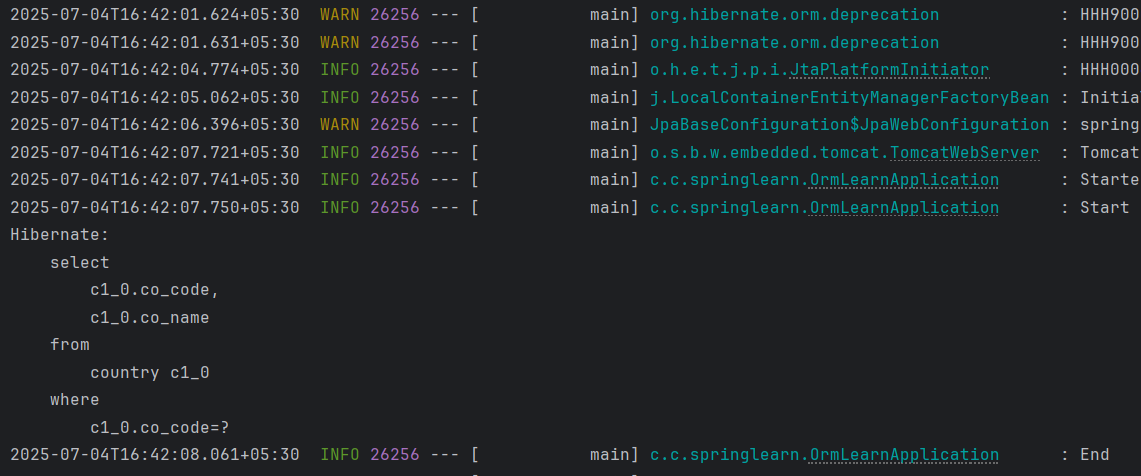
DELETE FROM country;  
  
INSERT INTO country (co\_code, co\_name) VALUES ('AF', 'Afghanistan');  
INSERT INTO country (co\_code, co\_name) VALUES ('AL', 'Albania');  
INSERT INTO country (co\_code, co\_name) VALUES ('DZ', 'Algeria');  
INSERT INTO country (co\_code, co\_name) VALUES ('AS', 'American Samoa');  
INSERT INTO country (co\_code, co\_name) VALUES ('AD', 'Andorra');  
INSERT INTO country (co\_code, co\_name) VALUES ('AO', 'Angola');  
INSERT INTO country (co\_code, co\_name) VALUES ('AI', 'Anguilla');  
INSERT INTO country (co\_code, co\_name) VALUES ('AQ', 'Antarctica');  
INSERT INTO country (co\_code, co\_name) VALUES ('AG', 'Antigua and Barbuda');  
INSERT INTO country (co\_code, co\_name) VALUES ('AR', 'Argentina');  
  
INSERT INTO country (co\_code, co\_name) VALUES ('AE', 'United Arab Emirates');  
INSERT INTO country (co\_code, co\_name) VALUES ('GB', 'United Kingdom');  
INSERT INTO country (co\_code, co\_name) VALUES ('US', 'United States');  
INSERT INTO country (co\_code, co\_name) VALUES ('UM', 'United States Minor Outlying Islands');  
INSERT INTO country (co\_code, co\_name) VALUES ('UY', 'Uruguay');  
INSERT INTO country (co\_code, co\_name) VALUES ('UZ', 'Uzbekistan');  
INSERT INTO country (co\_code, co\_name) VALUES ('VU', 'Vanuatu');  
INSERT INTO country (co\_code, co\_name) VALUES ('VE', 'Venezuela, Bolivarian Republic of');  
INSERT INTO country (co\_code, co\_name) VALUES ('VN', 'Viet Nam');  
INSERT INTO country (co\_code, co\_name) VALUES ('VG', 'Virgin Islands, British');  
INSERT INTO country (co\_code, co\_name) VALUES ('VI', 'Virgin Islands, U.S.');  
INSERT INTO country (co\_code, co\_name) VALUES ('WF', 'Wallis and Futuna');  
INSERT INTO country (co\_code, co\_name) VALUES ('EH', 'Western Sahara');  
INSERT INTO country (co\_code, co\_name) VALUES ('YE', 'Yemen');  
INSERT INTO country (co\_code, co\_name) VALUES ('ZM', 'Zambia');  
INSERT INTO country (co\_code, co\_name) VALUES ('ZW', 'Zimbabwe');

CountryApplication.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.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@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);  
 CountryService countryService = context.getBean(CountryService.class);  
  
 *getAllCountriesTest*(countryService); *testAddCountry*(countryService);

}  
  
  
 private static void getAllCountriesTest(CountryService countryService) {  
 *LOGGER*.info("Start");  
  
 try {  
 Country country = countryService.findByCode("ZW");  
 *LOGGER*.debug("Country:{}", country);  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Exception: {}", e.getMessage());  
 }  
  
 *LOGGER*.info("End");  
 }  
  
 private static void testAddCountry(CountryService countryService) {  
 *LOGGER*.info("Start");  
  
 Country newCountry = new Country();  
 newCountry.setCoCode("JP");  
 newCountry.setCoName("Japan");  
  
 countryService.addCountry(newCountry);  
  
 try {  
 Country country = countryService.findByCode("JP");  
 *LOGGER*.debug("Country added: {}", country);  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Country not found after adding: {}", e.getMessage());  
 }  
  
 *LOGGER*.info("End");  
 }  
  
  
}

Output:



**Hands on 7**

**Add a new country** 

* Create new method in CountryService.

@Transactional

public void addCountry(Country country)

* Invoke save() method of repository to get the country added.

countryRepository.save(country)

* Include new testAddCountry() method in OrmLearnApplication. Perform steps below:
  + Create new instance of country with a new code and name
  + Call countryService.addCountry() passing the country created in the previous step.
  + Invoke countryService.findCountryByCode() passing the same code used when adding a new country
  + Check in the database if the country is added

Sol.

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>  
  
   
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.2.5</version>   
 <relativePath/>   
 </parent>  
  
 <groupId>com.cognizant</groupId>  
 <artifactId>spring-learn</artifactId>  
 <version>1.0.0</version>  
 <packaging>jar</packaging>  
 <name>spring-learn</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>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <version>1.18.30</version>   
 <scope>provided</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
</project>

Application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/springlearn  
spring.datasource.username=root1  
spring.datasource.password=0Mg@np@t@yn@m@h@  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.format\_sql=true  
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect  
  
logging.level.org.springframework=INFO  
logging.level.com.cognizant.spring-learn=DEBUG

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 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;  
 }  
}

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 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.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/countries")  
public class CountryController {  
  
 @Autowired  
 private CountryService countryService;  
  
 @GetMapping("/{code}")  
 public ResponseEntity<Country> getCountry(@PathVariable String code) throws CountryNotFoundException {  
 return ResponseEntity.*ok*(countryService.findByCode(code));  
 }  
  
  
 @PostMapping  
 public ResponseEntity<Country> createCountry(@RequestBody Country country) {  
 return ResponseEntity.*ok*(countryService.addCountry(country));  
 }  
  
 @PutMapping  
 public ResponseEntity<Country> updateCountry(@RequestBody Country country) {  
 return ResponseEntity.*ok*(countryService.updateCountry(country));  
 }  
  
 @DeleteMapping("/{code}")  
 public ResponseEntity<Void> deleteCountry(@PathVariable String code) {  
 countryService.deleteCountry(code);  
 return ResponseEntity.*noContent*().build();  
 }  
  
 @GetMapping("/search")  
 public ResponseEntity<List<Country>> searchCountries(@RequestParam String name) {  
 return ResponseEntity.*ok*(countryService.findByPartialName(name));  
 }  
}

CountryService.java

package com.cognizant.springlearn.service;  
  
import com.cognizant.springlearn.model.Country;  
import com.cognizant.springlearn.service.exception.CountryNotFoundException;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface CountryService {  
 Country findByCode(String code) throws CountryNotFoundException;  
 Country addCountry(Country country);  
 Country updateCountry(Country country);  
 void deleteCountry(String code);  
 List<Country> findByPartialName(String name);  
}

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 findByCode(String countryCode) throws CountryNotFoundException {  
 Optional<Country> result = countryRepository.findById(countryCode);  
 if (!result.isPresent()) {  
 throw new CountryNotFoundException("Country with code " + countryCode + " not found");  
 }  
 return result.get();  
 }  
  
  
// @Override  
// public Optional<Country> findByCode(String code) {  
// return countryRepository.findById(code);  
// }  
  
 @Override  
 public Country addCountry(Country country) {  
 return countryRepository.save(country);  
 }  
  
 @Override  
 public Country updateCountry(Country country) {  
 return countryRepository.save(country);  
 }  
  
 @Override  
 public void deleteCountry(String code) {  
 countryRepository.deleteById(code);  
 }  
  
 @Override  
 public List<Country> findByPartialName(String name) {  
 return countryRepository.findByCoNameContainingIgnoreCase(name);  
 }  
}

data.sql

DELETE FROM country;  
  
INSERT INTO country (co\_code, co\_name) VALUES ('AF', 'Afghanistan');  
INSERT INTO country (co\_code, co\_name) VALUES ('AL', 'Albania');  
INSERT INTO country (co\_code, co\_name) VALUES ('DZ', 'Algeria');  
INSERT INTO country (co\_code, co\_name) VALUES ('AS', 'American Samoa');  
INSERT INTO country (co\_code, co\_name) VALUES ('AD', 'Andorra');  
INSERT INTO country (co\_code, co\_name) VALUES ('AO', 'Angola');  
INSERT INTO country (co\_code, co\_name) VALUES ('AI', 'Anguilla');  
INSERT INTO country (co\_code, co\_name) VALUES ('AQ', 'Antarctica');  
INSERT INTO country (co\_code, co\_name) VALUES ('AG', 'Antigua and Barbuda');  
INSERT INTO country (co\_code, co\_name) VALUES ('AR', 'Argentina');  
  
INSERT INTO country (co\_code, co\_name) VALUES ('AE', 'United Arab Emirates');  
INSERT INTO country (co\_code, co\_name) VALUES ('GB', 'United Kingdom');  
INSERT INTO country (co\_code, co\_name) VALUES ('US', 'United States');  
INSERT INTO country (co\_code, co\_name) VALUES ('UM', 'United States Minor Outlying Islands');  
INSERT INTO country (co\_code, co\_name) VALUES ('UY', 'Uruguay');  
INSERT INTO country (co\_code, co\_name) VALUES ('UZ', 'Uzbekistan');  
INSERT INTO country (co\_code, co\_name) VALUES ('VU', 'Vanuatu');  
INSERT INTO country (co\_code, co\_name) VALUES ('VE', 'Venezuela, Bolivarian Republic of');  
INSERT INTO country (co\_code, co\_name) VALUES ('VN', 'Viet Nam');  
INSERT INTO country (co\_code, co\_name) VALUES ('VG', 'Virgin Islands, British');  
INSERT INTO country (co\_code, co\_name) VALUES ('VI', 'Virgin Islands, U.S.');  
INSERT INTO country (co\_code, co\_name) VALUES ('WF', 'Wallis and Futuna');  
INSERT INTO country (co\_code, co\_name) VALUES ('EH', 'Western Sahara');  
INSERT INTO country (co\_code, co\_name) VALUES ('YE', 'Yemen');  
INSERT INTO country (co\_code, co\_name) VALUES ('ZM', 'Zambia');  
INSERT INTO country (co\_code, co\_name) VALUES ('ZW', 'Zimbabwe');

CountryApplication.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.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@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);  
 CountryService countryService = context.getBean(CountryService.class);  
  
 *getAllCountriesTest*(countryService); *testAddCountry*(countryService);

}  
  
  
 private static void getAllCountriesTest(CountryService countryService) {  
 *LOGGER*.info("Start");  
  
 try {  
 Country country = countryService.findByCode("ZW");  
 *LOGGER*.debug("Country:{}", country);  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Exception: {}", e.getMessage());  
 }  
  
 *LOGGER*.info("End");  
 }  
  
 private static void testAddCountry(CountryService countryService) {  
 *LOGGER*.info("Start");  
  
 Country newCountry = new Country();  
 newCountry.setCoCode("JP");  
 newCountry.setCoName("Japan");  
  
 countryService.addCountry(newCountry);  
  
 try {  
 Country country = countryService.findByCode("JP");  
 *LOGGER*.debug("Country added: {}", country);  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Country not found after adding: {}", e.getMessage());  
 }  
  
 *LOGGER*.info("End");  
 }  
  
  
}

Output:

