|  |  |
| --- | --- |
| **Week 3 – Spring Core and Maven,**  **ing Spring Data and JPA with Spring Boot,Hibernate** | **Su SupersetId-6431499**  **Name : Balaji V** |

**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.

**Code:**

**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.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<maven.compiler.source>23</maven.compiler.source>

<maven.compiler.target>23</maven.compiler.target>

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

<spring.version>6.1.2</spring.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.11.0</version>

<configuration>

<source>17</source>

<target>17</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**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.library.repository.BookRepository">

</bean>

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

</bean>

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

<constructor-arg ref="bookRepository"/>

</bean>

</beans>

**BookRepository.java:**

package com.library.repository;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class BookRepository{

private Map<Integer, String> books;

private int bookIdCounter;

public BookRepository(){

this.books=new HashMap<>();

this.bookIdCounter=1;

initializeSampleData();

System.out.println("\*BookRepository instance created and initialized");

}

private void initializeSampleData(){

books.put(bookIdCounter++, "The Great Gatsby by F. Scott Fitzgerald");

books.put(bookIdCounter++, "To Kill a Mockingbird by Harper Lee");

books.put(bookIdCounter++, "1984 by George Orwell");

books.put(bookIdCounter++, "Wings of Fire by A.P.J. Abdul Kalam");

System.out.println("\*Sample book data initialized in repository");

}

public int addBook(String bookTitle){

int newBookId=bookIdCounter++;

books.put(newBookId, bookTitle);

System.out.println("Book added to repository: ID=" + newBookId + ", Title=" + bookTitle);

return newBookId;

}

public String getBookById(int bookId){

String book=books.get(bookId);

if(book!=null){

System.out.println("Book retrieved from repository: ID=" + bookId + ", Title=" + book);

}else{

System.out.println("Book not found in repository with ID: " + bookId);

}

return book;

}

public List<String> getAllBooks(){

List<String> allBooks=new ArrayList<>();

for(Map.Entry<Integer, String> entry : books.entrySet()){

allBooks.add("ID: " + entry.getKey() + " - " + entry.getValue());

}

System.out.println("Retrieved all books from repository. Total count: " + allBooks.size());

return allBooks;

}

public boolean removeBook(int bookId){

String removedBook=books.remove(bookId);

if(removedBook!=null){

System.out.println("Book removed from repository: ID=" + bookId + ", Title=" + removedBook);

return true;

}else{

System.out.println("Cannot remove book - not found in repository with ID: " + bookId);

return false;

}

}

public List<String> searchBooks(String searchTerm){

List<String> matchingBooks=new ArrayList<>();

for(Map.Entry<Integer, String> entry : books.entrySet()){

if(entry.getValue().toLowerCase().contains(searchTerm.toLowerCase())){

matchingBooks.add("ID: " + entry.getKey() + " - " + entry.getValue());

}

}

System.out.println("Search completed in repository for term: '" + searchTerm + "'. Found " + matchingBooks.size() + " matching books");

return matchingBooks;

}

public int getTotalBooks(){

int total=books.size();

System.out.println("Total books in repository: " + total);

return total;

}

}

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

import java.util.List;

public class BookService {

private BookRepository bookRepository;

public BookService(){

System.out.println("\*BookService instance created");

}

public BookService(BookRepository bookRepository){

this.bookRepository=bookRepository;

System.out.println("\*BookService instance created with BookRepository dependency");

}

public void setBookRepository(BookRepository bookRepository){

this.bookRepository=bookRepository;

System.out.println("\*BookRepository dependency injected into BookService");

}

public BookRepository getBookRepository(){

return bookRepository;

}

public int addNewBook(String bookTitle){

System.out.println("BookService: Processing request to add new book - " + bookTitle);

if(bookTitle==null || bookTitle.trim().isEmpty()){

System.out.println("BookService: Invalid book title provided");

return -1;

}

int bookId=bookRepository.addBook(bookTitle);

System.out.println("BookService: Book successfully added with ID: " + bookId);

return bookId;

}

public String findBookById(int bookId){

System.out.println("BookService: Processing request to find book with ID: " + bookId);

if(bookId<=0){

System.out.println("BookService: Invalid book ID provided: " + bookId);

return null;

}

String book=bookRepository.getBookById(bookId);

if(book!=null){

System.out.println("BookService: Book found successfully");

}else{

System.out.println("BookService: Book not found");

}

return book;

}

public List<String> getAllBooks(){

System.out.println("BookService: Processing request to get all books");

List<String> books=bookRepository.getAllBooks();

System.out.println("BookService: Retrieved " + books.size() + " books");

return books;

}

public boolean removeBook(int bookId){

System.out.println("BookService: Processing request to remove book with ID: " + bookId);

if(bookId<=0){

System.out.println("BookService: Invalid book ID provided: " + bookId);

return false;

}

boolean removed=bookRepository.removeBook(bookId);

if(removed){

System.out.println("BookService: Book successfully removed");

}else{

System.out.println("BookService: Failed to remove book");

}

return removed;

}

public List<String> searchBooks(String searchTerm){

System.out.println("BookService: Processing search request for: " + searchTerm);

if (searchTerm==null || searchTerm.trim().isEmpty()){

System.out.println("BookService: Invalid search term provided");

return java.util.Collections.emptyList();

}

List<String> results=bookRepository.searchBooks(searchTerm);

System.out.println("BookService: Search completed, found " + results.size() + " results");

return results;

}

public int getLibraryStatistics(){

System.out.println("BookService: Processing request for library statistics");

int totalBooks=bookRepository.getTotalBooks();

System.out.println("BookService: Library contains " + totalBooks + " books");

return totalBooks;

}

public void displayLibraryInfo(){

System.out.println("\nCTS-LIBRARY MANAGEMENT SYSTEM!!!");

System.out.println("BookService Status: " + (bookRepository != null ? "Active" : "Inactive"));

System.out.println("Repository Status: " + (bookRepository != null ? "Connected" : "Disconnected"));

if(bookRepository!=null){

int totalBooks=getLibraryStatistics();

System.out.println("Total Books Available: " + totalBooks);

}

System.out.println();

}

public void performLibraryOperations(){

System.out.println("\nPERFORMING LIBRARY OPERATIONS!!!");

displayLibraryInfo();

System.out.println("Current books in library:");

List<String> allBooks=getAllBooks();

allBooks.forEach(book -> System.out.println(" " + book));

System.out.println("\nAdding a new book...");

int newBookId=addNewBook("The Catcher in the Rye by J.D. Salinger");

System.out.println("\nSearching for books containing 'Great'...");

List<String> searchResults=searchBooks("Great");

searchResults.forEach(book -> System.out.println(" " + book));

System.out.println("\nFinding book with ID 1...");

String foundBook=findBookById(1);

if(foundBook!=null){

System.out.println(" Found: " + foundBook);

}

System.out.println("\nLIBRARY OPERATIONS COMPLETED!!!\n");

}

}

**LibraryManagementApplication.java:**

package com.library;

import com.library.service.BookService;

import com.library.repository.BookRepository;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.Scanner;

import java.util.List;

public class LibraryManagementApplication{

private static Scanner scanner=new Scanner(System.in);

private static BookService bookService;

public static void main(String[] args){

System.out.println("CTS-LIBRARY MANAGEMENT SYSTEM - SPRING APPLICATION ");

try{

System.out.println("\*Loading Spring Application Context...");

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

System.out.println("\*Spring Application Context loaded successfully!");

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

runInteractiveLibrarySystem();

System.out.println("APPLICATION COMPLETED SUCCESSFULLY!!!");

}catch(Exception e){

System.err.println("Error occurred while running the application:");

e.printStackTrace();

}finally{

scanner.close();

}

}

private static void runInteractiveLibrarySystem(){

bookService.displayLibraryInfo();

boolean running=true;

while(running){

displayMenu();

int choice=getUserChoice();

switch(choice){

case 1:

viewAllBooks();

break;

case 2:

addNewBook();

break;

case 3:

searchBooks();

break;

case 4:

findBookById();

break;

case 5:

removeBook();

break;

case 6:

getLibraryStatistics();

break;

case 7:

running=false;

System.out.println("Thank you for using Library Management System!");

break;

default:

System.out.println("Invalid choice. Please try again.");

}

if(running){

System.out.println("\nPress Enter to continue...");

scanner.nextLine();

}

}

}

private static void displayMenu(){

System.out.println("\nCTS-LIBRARY MENU...");

System.out.println("1. View All Books");

System.out.println("2. Add New Book");

System.out.println("3. Search Books");

System.out.println("4. Find Book by ID");

System.out.println("5. Remove Book");

System.out.println("6. Library Statistics");

System.out.println("7. Exit");

System.out.print("Enter your choice: ");

}

private static int getUserChoice(){

try{

int choice=Integer.parseInt(scanner.nextLine());

return choice;

}catch(NumberFormatException e){

return -1;

}

}

private static void viewAllBooks(){

System.out.println("\nALL BOOKS...");

List<String> books=bookService.getAllBooks();

if(books.isEmpty()){

System.out.println("No books found in the library.");

}else{

books.forEach(book -> System.out.println(" " + book));

}

}

private static void addNewBook(){

System.out.println("\nADD NEW BOOK");

System.out.print("Enter book title: ");

String title=scanner.nextLine();

if(title.trim().isEmpty()){

System.out.println("Book title cannot be empty.");

return;

}

int bookId=bookService.addNewBook(title);

if(bookId>0) {

System.out.println("Book added successfully with ID: " + bookId);

}else{

System.out.println("Failed to add book.");

}

}

private static void searchBooks(){

System.out.println("\nSEARCH BOOKS");

System.out.print("Enter search term: ");

String searchTerm=scanner.nextLine();

if(searchTerm.trim().isEmpty()){

System.out.println("Search term cannot be empty.");

return;

}

List<String> results=bookService.searchBooks(searchTerm);

if(results.isEmpty()){

System.out.println("No books found matching: " + searchTerm);

}else{

System.out.println("Search results:");

results.forEach(book -> System.out.println(" " + book));

}

}

private static void findBookById(){

System.out.println("\nFIND BOOK BY ID");

System.out.print("Enter book ID: ");

try{

int bookId=Integer.parseInt(scanner.nextLine());

String book=bookService.findBookById(bookId);

if(book!=null){

System.out.println("Book found: " + book);

}else{

System.out.println("Book not found with ID: " + bookId);

}

}catch(NumberFormatException e){

System.out.println("Invalid book ID. Please enter a valid number.");

}

}

private static void removeBook(){

System.out.println("\nREMOVE BOOK");

System.out.print("Enter book ID to remove: ");

try{

int bookId=Integer.parseInt(scanner.nextLine());

boolean removed=bookService.removeBook(bookId);

if(removed){

System.out.println("Book removed successfully.");

}else{

System.out.println("Failed to remove book with ID: " + bookId);

}

}catch(NumberFormatException e){

System.out.println("Invalid book ID. Please enter a valid number.");

}

}

private static void getLibraryStatistics(){

System.out.println("\nLIBRARY STATISTICS");

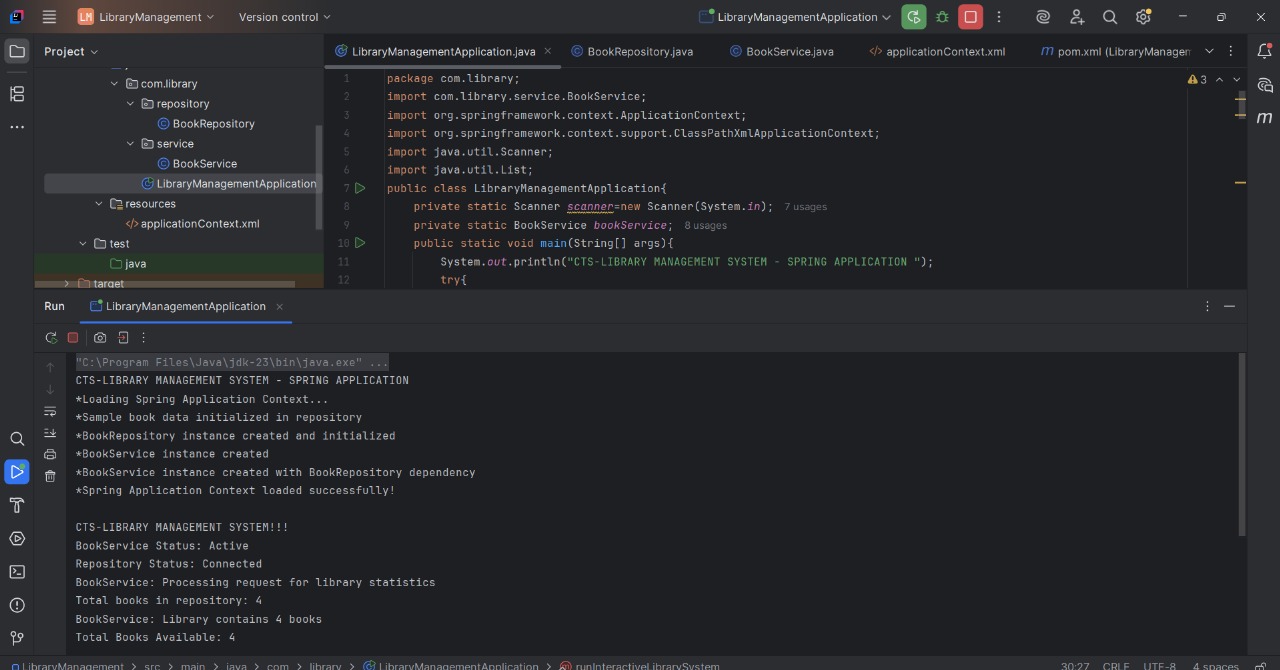
int totalBooks=bookService.getLibraryStatistics();

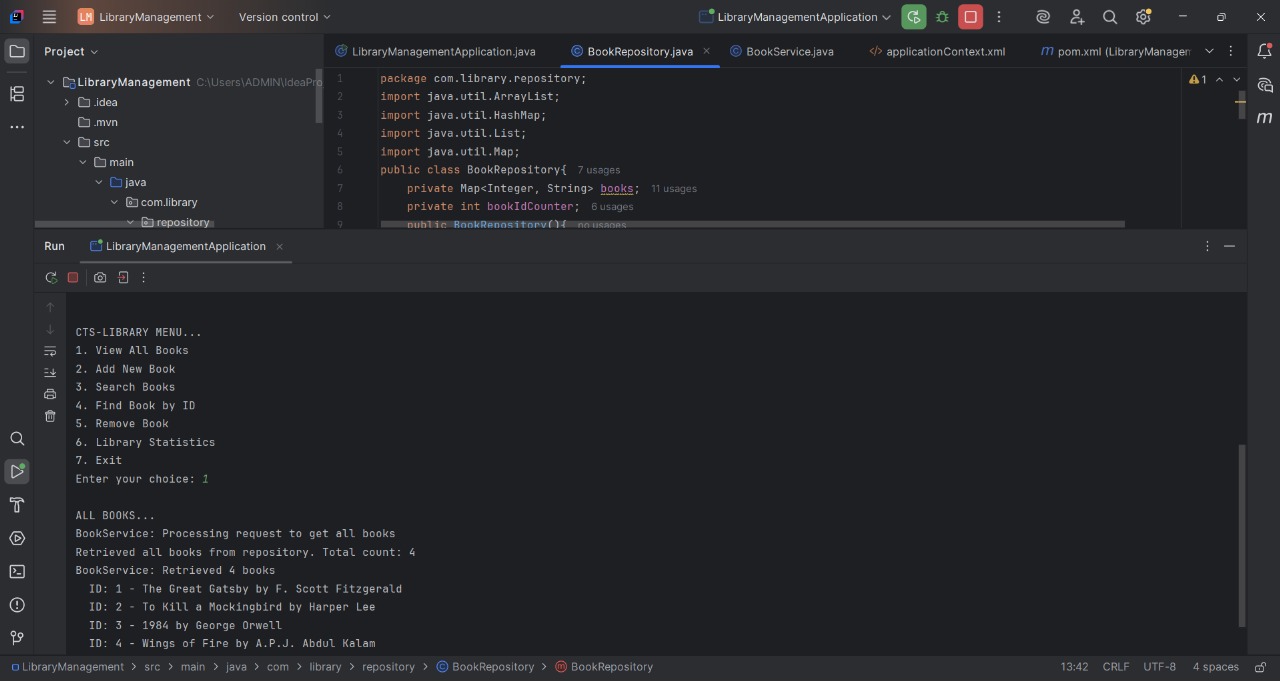
System.out.println("Total books in library: " + totalBooks);

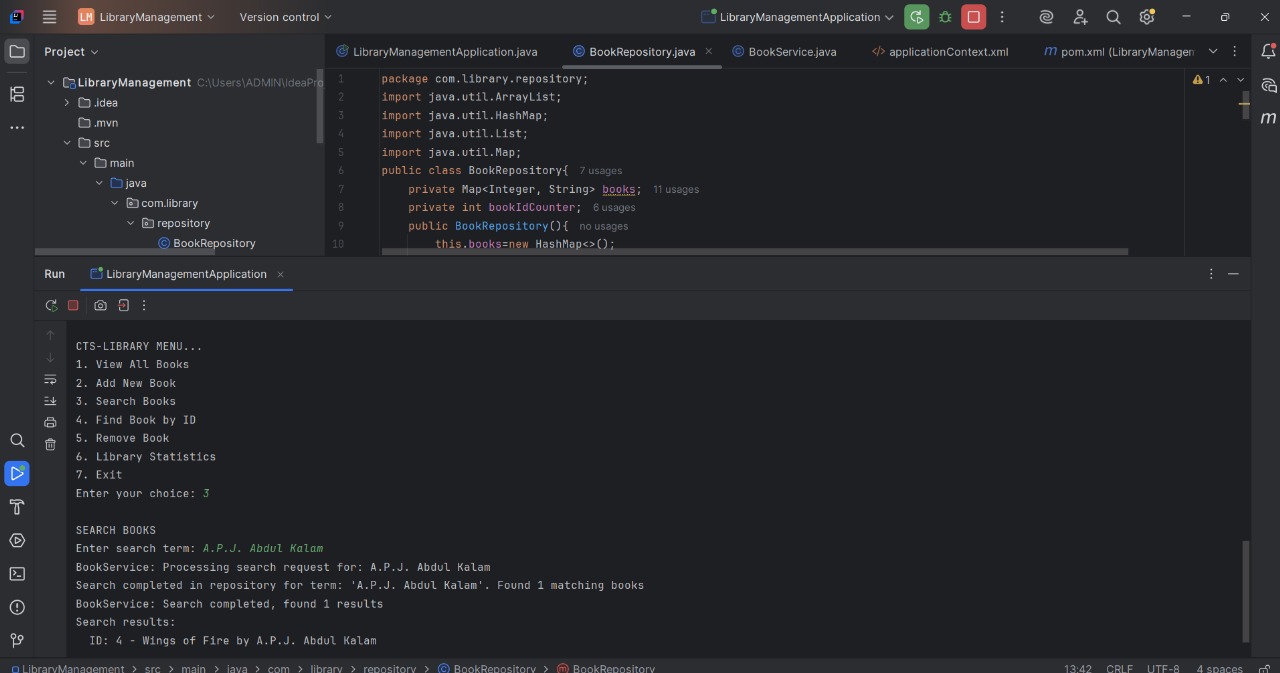
}

}

**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.

**Code:**

**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.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<maven.compiler.source>23</maven.compiler.source>

<maven.compiler.target>23</maven.compiler.target>

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

<spring.version>6.1.2</spring.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.11.0</version>

<configuration>

<source>17</source>

<target>17</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**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.library.repository.BookRepository">

</bean>

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

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

</bean>

</beans>

**Book.java:**

package com.library;

public class Book{

private Long id;

private String title;

private String author;

private String isbn;

private boolean available;

public Book(){

}

public Book(Long id, String title, String author, String isbn, boolean available){

this.id=id;

this.title=title;

this.author=author;

this.isbn=isbn;

this.available=available;

}

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 getIsbn(){

return isbn;

}

public void setIsbn(String isbn){

this.isbn=isbn;

}

public boolean isAvailable(){

return available;

}

public void setAvailable(boolean available){

this.available=available;

}

@Override

public String toString(){

return "Book{" + "id=" + id + ", title='" + title + '\'' + ", author='" + author + '\'' + ", isbn='" + isbn + '\'' + ", available=" + available + '}';

}

}

**BookRepository.java:**

package com.library.repository;

import com.library.Book;

import java.util.\*;

public class BookRepository{

private Map<Long,Book> books;

private Long nextId;

public BookRepository(){

this.books=new HashMap<>();

this.nextId=1L;

initializeSampleData();

}

private void initializeSampleData(){

addBook(new Book(nextId++, "The Lord of the Rings", "J.R.R. Tolkien", "978-0-7432-7356-5", true));

addBook(new Book(nextId++, "To Kill a Mockingbird", "Harper Lee", "978-0-06-112008-4", true));

addBook(new Book(nextId++, "Wings of Fire", "A.P.J. Abdul Kalam", "978-0-452-28423-4", false));

addBook(new Book(nextId++, "The Jungle Book", "Rudyard Kipling", "978-0-14-143951-8", true));

}

public List<Book> findAll(){

System.out.println("BookRepository: Retrieving all books");

return new ArrayList<>(books.values());

}

public Book findById(Long id){

System.out.println("BookRepository: Finding book with ID: " + id);

return books.get(id);

}

public Book save(Book book){

if (book.getId()==null){

book.setId(nextId++);

}

books.put(book.getId(), book);

System.out.println("BookRepository: Saved book - " + book.getTitle());

return book;

}

public void addBook(Book book){

books.put(book.getId(),book);

}

public void deleteById(Long id){

Book removedBook=books.remove(id);

if(removedBook!=null){

System.out.println("BookRepository: Deleted book - " + removedBook.getTitle());

}else{

System.out.println("BookRepository: Book with ID " + id + " not found");

}

}

public List<Book> findByAuthor(String author){

System.out.println("BookRepository: Finding books by author: " + author);

return books.values().stream().filter(book -> book.getAuthor().toLowerCase().contains(author.toLowerCase())).collect(ArrayList::new, ArrayList::add, ArrayList::addAll);

}

public List<Book> findAvailableBooks(){

System.out.println("BookRepository: Finding available books");

return books.values().stream().filter(Book::isAvailable).collect(ArrayList::new, ArrayList::add, ArrayList::addAll);

}

public int getTotalBooks(){

return books.size();

}

}

**BookService.java:**

package com.library.service;

import com.library.Book;

import com.library.repository.BookRepository;

import java.util.List;

public class BookService{

private BookRepository bookRepository;

public BookService(){

}

public BookService(BookRepository bookRepository){

this.bookRepository=bookRepository;

}

public void setBookRepository(BookRepository bookRepository){

this.bookRepository=bookRepository;

System.out.println("\*BookRepository injected into BookService");

}

public BookRepository getBookRepository(){

return bookRepository;

}

public List<Book> getAllBooks(){

System.out.println("BookService: Retrieving all books...");

List<Book> books=bookRepository.findAll();

System.out.println("Found " + books.size() + " books in the library");

return books;

}

public Book getBookById(Long id){

System.out.println("BookService: Searching for book with ID: " + id);

Book book=bookRepository.findById(id);

if(book==null){

System.out.println("Book with ID " + id + " not found");

}else{

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

}

return book;

}

public Book addBook(Book book){

System.out.println("BookService: Adding new book - " + book.getTitle());

try{

if(book.getTitle()==null || book.getTitle().trim().isEmpty()){

throw new IllegalArgumentException("Book title cannot be empty");

}

if(book.getAuthor()==null || book.getAuthor().trim().isEmpty()){

throw new IllegalArgumentException("Book author cannot be empty");

}

if(book.getIsbn()==null || book.getIsbn().trim().isEmpty()){

throw new IllegalArgumentException("Book ISBN cannot be empty");

}

Book savedBook=bookRepository.save(book);

System.out.println("Book added successfully with ID: " + savedBook.getId());

return savedBook;

}catch(Exception e){

System.out.println("Error adding book: " + e.getMessage());

throw e;

}

}

public Book updateBook(Book book){

System.out.println("BookService: Updating book - " + book.getTitle());

try{

Book existingBook=bookRepository.findById(book.getId());

if(existingBook==null){

throw new IllegalArgumentException("Book with ID " + book.getId() + " not found");

}

Book updatedBook= bookRepository.save(book);

System.out.println("Book updated successfully");

return updatedBook;

}catch(Exception e){

System.out.println("Error updating book: " + e.getMessage());

throw e;

}

}

public void deleteBook(Long id){

System.out.println("BookService: Deleting book with ID: " + id);

try{

Book book=bookRepository.findById(id);

if(book==null){

throw new IllegalArgumentException("Book with ID " + id + " not found");

}

bookRepository.deleteById(id);

System.out.println("Book deleted successfully");

}catch(Exception e){

System.out.println("Error deleting book: " + e.getMessage());

throw e;

}

}

public List<Book> getBooksByAuthor(String author){

System.out.println("BookService: Searching books by author: " + author);

try{

if(author==null || author.trim().isEmpty()){

throw new IllegalArgumentException("Author name cannot be empty");

}

List<Book> books=bookRepository.findByAuthor(author);

System.out.println("Found " + books.size() + " books by " + author);

return books;

}catch(Exception e){

System.out.println("Error searching books by author: " + e.getMessage());

throw e;

}

}

public List<Book> getAvailableBooks(){

System.out.println("BookService: Retrieving available books...");

List<Book> books=bookRepository.findAvailableBooks();

System.out.println("Found " + books.size() + " available books");

return books;

}

public boolean borrowBook(Long bookId){

System.out.println("BookService: Processing borrow request for book ID: " + bookId);

try{

Book book=bookRepository.findById(bookId);

if(book==null){

System.out.println("Book with ID " + bookId + " not found");

return false;

}

if(!book.isAvailable()){

System.out.println("Book '" + book.getTitle() + "' is already borrowed");

return false;

}

book.setAvailable(false);

bookRepository.save(book);

System.out.println("Book '" + book.getTitle() + "' borrowed successfully");

return true;

}catch(Exception e){

System.out.println("Error borrowing book: " + e.getMessage());

return false;

}

}

public boolean returnBook(Long bookId){

System.out.println("BookService: Processing return request for book ID: " + bookId);

try{

Book book=bookRepository.findById(bookId);

if(book==null){

System.out.println("Book with ID " + bookId + " not found");

return false;

}

if(book.isAvailable()){

System.out.println("Book '" + book.getTitle() + "' is already available (not borrowed)");

return false;

}

book.setAvailable(true);

bookRepository.save(book);

System.out.println("Book '" + book.getTitle() + "' returned successfully");

return true;

}catch(Exception e){

System.out.println("Error returning book: " + e.getMessage());

return false;

}

}

public int getTotalBooksCount(){

System.out.println("BookService: Counting total books...");

int count=bookRepository.getTotalBooks();

System.out.println("Total books count: " + count);

return count;

}

public void displayLibraryStatistics(){

System.out.println("\n" + "=".repeat(50));

System.out.println("LIBRARY STATISTICS");

System.out.println("=".repeat(50));

int totalBooks=getTotalBooksCount();

List<Book> availableBooks=getAvailableBooks();

int availableCount=availableBooks.size();

int borrowedCount=totalBooks-availableCount;

System.out.println("Total Books: " + totalBooks);

System.out.println("Available Books: " + availableCount);

System.out.println("Borrowed Books: " + borrowedCount);

if(totalBooks>0){

double availabilityRate=(double) availableCount / totalBooks \* 100;

System.out.printf("Availability Rate: %.1f%%\n", availabilityRate);

}

System.out.println("Library Status: " + (availableCount > 0 ? "Open for borrowing" : "No books available"));

System.out.println("=".repeat(50));

}

public List<Book> searchBooksByTitle(String title){

System.out.println("BookService: Searching books by title: " + title);

try{

if(title==null || title.trim().isEmpty()){

throw new IllegalArgumentException("Title cannot be empty");

}

List<Book> allBooks=bookRepository.findAll();

List<Book> matchingBooks=allBooks.stream()

.filter(book -> book.getTitle().toLowerCase().contains(title.toLowerCase()))

.collect(java.util.stream.Collectors.toList());

System.out.println("Found " + matchingBooks.size() + " books matching title: " + title);

return matchingBooks;

}catch(Exception e){

System.out.println("Error searching books by title: " + e.getMessage());

throw e;

}

}

public List<Book> getOverdueBooks(){

System.out.println("BookService: Retrieving overdue books...");

List<Book> borrowedBooks=bookRepository.findAll().stream()

.filter(book -> !book.isAvailable())

.collect(java.util.stream.Collectors.toList());

System.out.println("Found " + borrowedBooks.size() + " potentially overdue books");

return borrowedBooks;

}

private boolean validateBook(Book book){

if (book==null){

System.out.println("Book cannot be null");

return false;

}

if(book.getTitle()==null || book.getTitle().trim().isEmpty()){

System.out.println("Book title is required");

return false;

}

if(book.getAuthor()==null || book.getAuthor().trim().isEmpty()){

System.out.println("Book author is required");

return false;

}

if(book.getIsbn()==null || book.getIsbn().trim().isEmpty()){

System.out.println("Book ISBN is required");

return false;

}

return true;

}}

**LibraryManagementApplication.java:**

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

import java.util.Scanner;

public class LibraryManagementApplication{

private static BookService bookService;

private static Scanner scanner;

public static void main(String[] args){

System.out.println("Welcome to Library Management System!!!");

System.out.println("Loading Spring Context...");

try{

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

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

scanner=new Scanner(System.in);

System.out.println("\*Spring Context Loaded Successfully!");

System.out.println("\*BookService bean initialized");

System.out.println("\*BookRepository dependency injected");

runInteractiveMenu();

}catch(Exception e){

System.err.println("Error initializing Spring context: " + e.getMessage());

e.printStackTrace();

}finally{

if(scanner!=null){

scanner.close();

}

}

}

private static void runInteractiveMenu(){

while(true){

displayMenu();

try{

System.out.print("Enter your choice (1-10): ");

int choice=Integer.parseInt(scanner.nextLine().trim());

switch(choice){

case 1:

viewAllBooks();

break;

case 2:

viewAvailableBooks();

break;

case 3:

searchBookById();

break;

case 4:

searchBooksByAuthor();

break;

case 5:

addNewBook();

break;

case 6:

borrowBook();

break;

case 7:

returnBook();

break;

case 8:

deleteBook();

break;

case 9:

viewLibraryStatistics();

break;

case 10:

System.out.println("\nThank you for using Library Management System!");

return;

default:

System.out.println("Invalid choice! Please enter a number between 1-10.");

}

System.out.println("\nPress Enter to continue...");

scanner.nextLine();

}catch(NumberFormatException e){

System.out.println("Invalid input! Please enter a valid number.");

System.out.println("Press Enter to continue...");

scanner.nextLine();

}catch(Exception e){

System.out.println("An error occurred: " + e.getMessage());

System.out.println("Press Enter to continue...");

scanner.nextLine();

}

}

}

private static void displayMenu(){

System.out.println("\n" + "=".repeat(50));

System.out.println("CTS-LIBRARY MANAGEMENT SYSTEM!!!");

System.out.println("=".repeat(50));

System.out.println("1.View All Books");

System.out.println("2.View Available Books");

System.out.println("3.Search Book by ID");

System.out.println("4.Search Books by Author");

System.out.println("5.Add New Book");

System.out.println("6.Borrow Book");

System.out.println("7.Return Book");

System.out.println("8.Delete Book");

System.out.println("9.View Library Statistics");

System.out.println("10.Exit");

System.out.println("=".repeat(50));

}

private static void viewAllBooks(){

System.out.println("\nALL BOOKS IN LIBRARY:");

System.out.println("-".repeat(80));

List<Book> books=bookService.getAllBooks();

if(books.isEmpty()){

System.out.println("No books found in the library.");

}else{

System.out.printf("%-5s %-25s %-20s %-20s %-10s%n", "ID", "Title", "Author", "ISBN", "Status");

System.out.println("-".repeat(80));

for(Book book : books){

String status=book.isAvailable() ? "Available" : "Borrowed";

System.out.printf("%-5d %-25s %-20s %-20s %-10s%n", book.getId(),truncateString(book.getTitle(), 24), truncateString(book.getAuthor(), 19), book.getIsbn(), status);

}

}

}

private static void viewAvailableBooks(){

System.out.println("\nAVAILABLE BOOKS:");

System.out.println("-".repeat(80));

List<Book> books=bookService.getAvailableBooks();

if(books.isEmpty()){

System.out.println("No available books found.");

}else{

System.out.printf("%-5s %-25s %-20s %-20s%n", "ID", "Title", "Author", "ISBN");

System.out.println("-".repeat(80));

for(Book book : books){

System.out.printf("%-5d %-25s %-20s %-20s%n", book.getId(),truncateString(book.getTitle(), 24), truncateString(book.getAuthor(), 19), book.getIsbn());

}

}

}

private static void searchBookById(){

System.out.print("\nEnter Book ID to search: ");

try{

Long id=Long.parseLong(scanner.nextLine().trim());

Book book=bookService.getBookById(id);

if(book!=null){

System.out.println("\nBOOK FOUND:");

System.out.println("-".repeat(40));

System.out.println("ID: " + book.getId());

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

System.out.println("Author: " + book.getAuthor());

System.out.println("ISBN: " + book.getIsbn());

System.out.println("Status: " + (book.isAvailable() ? "Available" : "Borrowed"));

}else{

System.out.println("Book with ID " + id + " not found!");

}

}catch(NumberFormatException e){

System.out.println("Invalid ID format! Please enter a valid number.");

}

}

private static void searchBooksByAuthor(){

System.out.print("\nEnter Author name to search: ");

String author = scanner.nextLine().trim();

if (author.isEmpty()){

System.out.println("Author name cannot be empty!");

return;

}

List<Book> books=bookService.getBooksByAuthor(author);

if(books.isEmpty()){

System.out.println(" No books found by author: " + author);

}else{

System.out.println("\nBOOKS BY " + author.toUpperCase() + ":");

System.out.println("-".repeat(80));

System.out.printf("%-5s %-25s %-20s %-10s%n", "ID", "Title", "ISBN", "Status");

System.out.println("-".repeat(80));

for(Book book : books){

String status=book.isAvailable() ? "Available" : "Borrowed";

System.out.printf("%-5d %-25s %-20s %-10s%n", book.getId(), truncateString(book.getTitle(), 24), book.getIsbn(), status);

}

}

}

private static void addNewBook(){

System.out.println("\nADD NEW BOOK:");

System.out.println("-".repeat(40));

System.out.print("Enter Book Title: ");

String title=scanner.nextLine().trim();

System.out.print("Enter Author Name: ");

String author= scanner.nextLine().trim();

System.out.print("Enter ISBN: ");

String isbn=scanner.nextLine().trim();

System.out.print("Is book available? (y/n): ");

String availableInput=scanner.nextLine().trim().toLowerCase();

boolean available=availableInput.equals("y") || availableInput.equals("yes");

if(title.isEmpty() || author.isEmpty() || isbn.isEmpty()){

System.out.println("All fields are required!");

return;

}

Book newBook=new Book(null, title, author, isbn, available);

Book addedBook=bookService.addBook(newBook);

System.out.println("Book added successfully!");

System.out.println("Book ID: " + addedBook.getId());

System.out.println("Title: " + addedBook.getTitle());

System.out.println("Author: " + addedBook.getAuthor());

}

private static void borrowBook(){

System.out.print("\nEnter Book ID to borrow: ");

try{

Long id=Long.parseLong(scanner.nextLine().trim());

boolean success=bookService.borrowBook(id);

if(success){

Book book=bookService.getBookById(id);

System.out.println("Book borrowed successfully!");

System.out.println("You have borrowed: " + book.getTitle());

System.out.println("Author: " + book.getAuthor());

System.out.println("Please return within 15 days.");

}else{

System.out.println("Unable to borrow book. Book may not exist or is already borrowed.");

}

}catch(NumberFormatException e){

System.out.println("Invalid ID format! Please enter a valid number.");

}

}

private static void returnBook(){

System.out.print("\nEnter Book ID to return: ");

try{

Long id=Long.parseLong(scanner.nextLine().trim());

boolean success=bookService.returnBook(id);

if(success){

Book book=bookService.getBookById(id);

System.out.println("Book returned successfully!");

System.out.println("You have returned: " + book.getTitle());

System.out.println("Author: " + book.getAuthor());

System.out.println("Thank you for returning on time!");

}else{

System.out.println("Unable to return book. Book may not exist or is already available.");

}

}catch(NumberFormatException e){

System.out.println("Invalid ID format! Please enter a valid number.");

}

}

private static void deleteBook(){

System.out.print("\nEnter Book ID to delete: ");

try{

Long id=Long.parseLong(scanner.nextLine().trim());

Book book=bookService.getBookById(id);

if(book==null){

System.out.println("Book with ID " + id + " not found!");

return;

}

System.out.println("Book to delete: " + book.getTitle() + " by " + book.getAuthor());

System.out.print("Are you sure you want to delete this book? (y/n): ");

String confirm=scanner.nextLine().trim().toLowerCase();

if(confirm.equals("y") || confirm.equals("yes")){

bookService.deleteBook(id);

System.out.println("Book deleted successfully!");

}else{

System.out.println("Book deletion cancelled.");

}

}catch(NumberFormatException e){

System.out.println("Invalid ID format! Please enter a valid number.");

}

}

private static void viewLibraryStatistics(){

System.out.println("\nLIBRARY STATISTICS:");

System.out.println("-".repeat(40));

int totalBooks=bookService.getTotalBooksCount();

int availableBooks=bookService.getAvailableBooks().size();

int borrowedBooks=totalBooks-availableBooks;

System.out.println("Total Books: " + totalBooks);

System.out.println("Available Books: " + availableBooks);

System.out.println("Borrowed Books: " + borrowedBooks);

if(totalBooks>0){

double availabilityPercentage=(double) availableBooks / totalBooks \* 100;

System.out.printf("Availability Rate: %.1f%%\n", availabilityPercentage);

}

System.out.println("-".repeat(40));

System.out.println("Library Status: " + (availableBooks > 0 ? "Open for borrowing" : "No books available"));

}

private static String truncateString(String str, int maxLength){

if (str.length()<=maxLength){

return str;

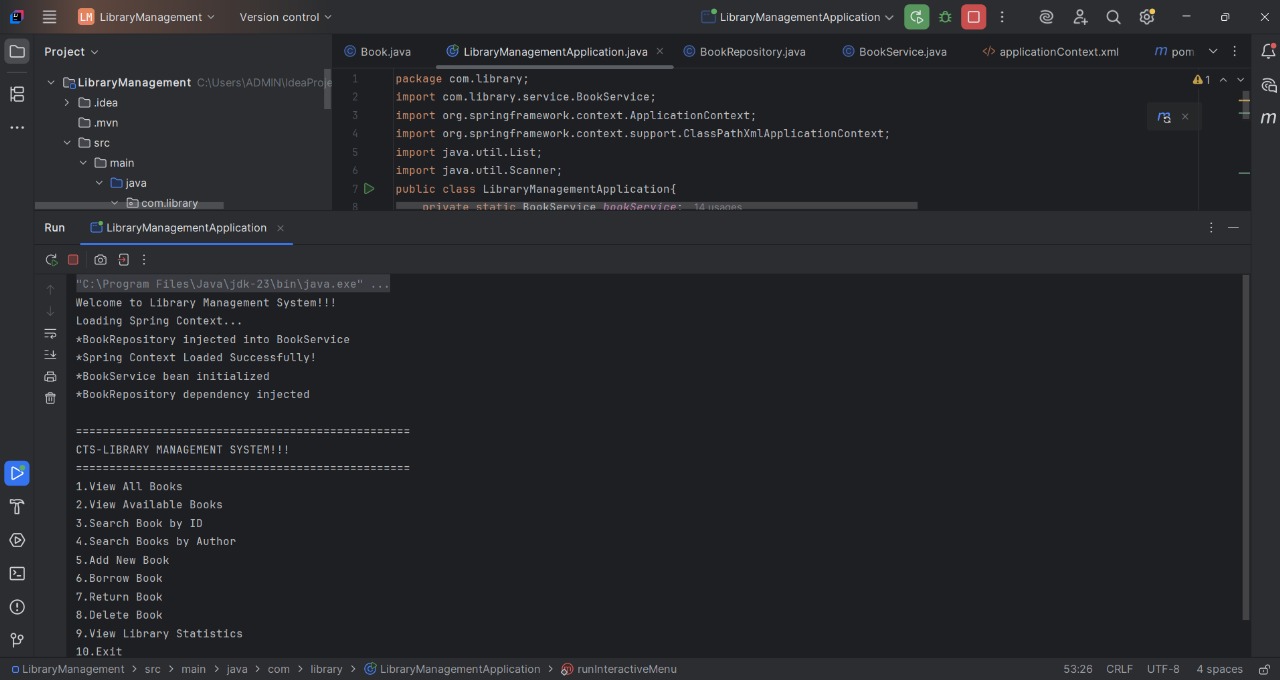
}

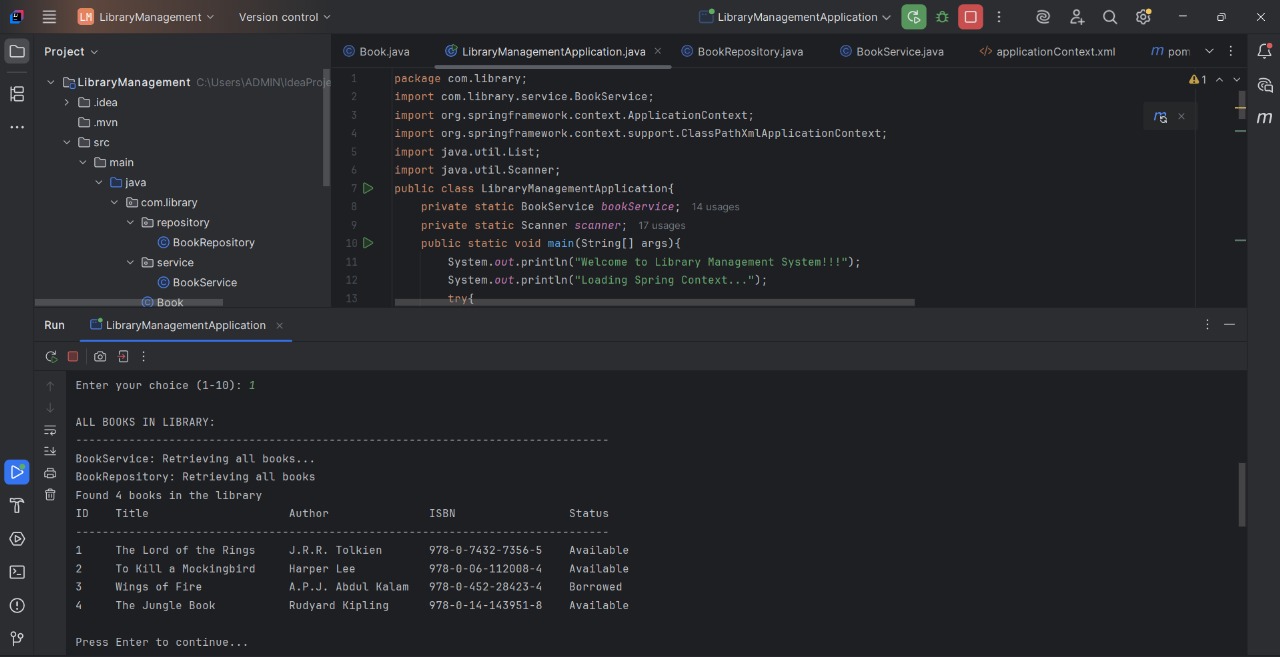
return str.substring(0, maxLength - 3) + "...";

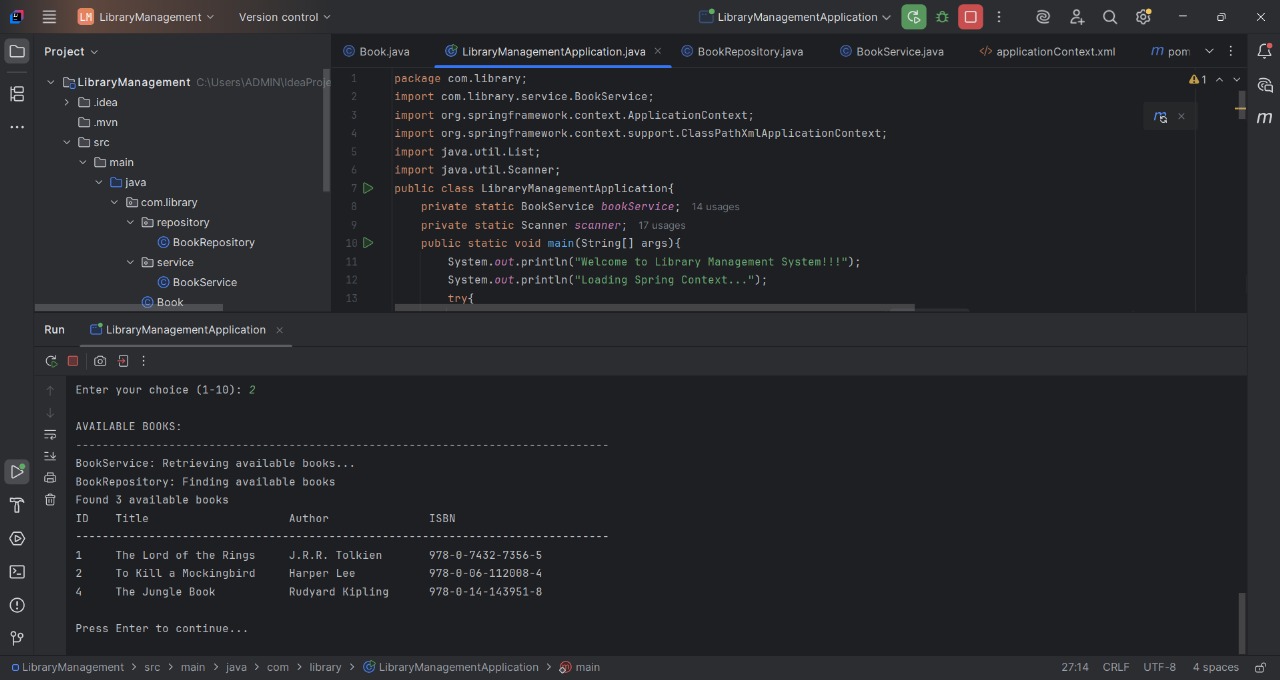
}

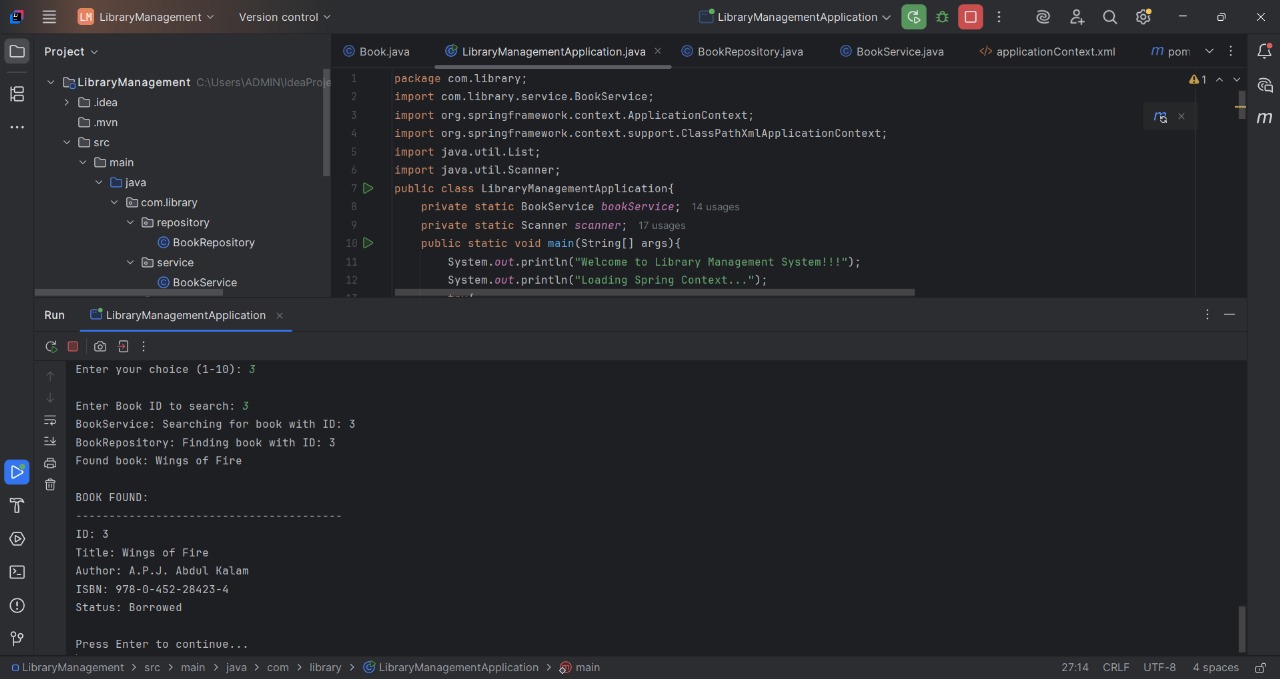
}

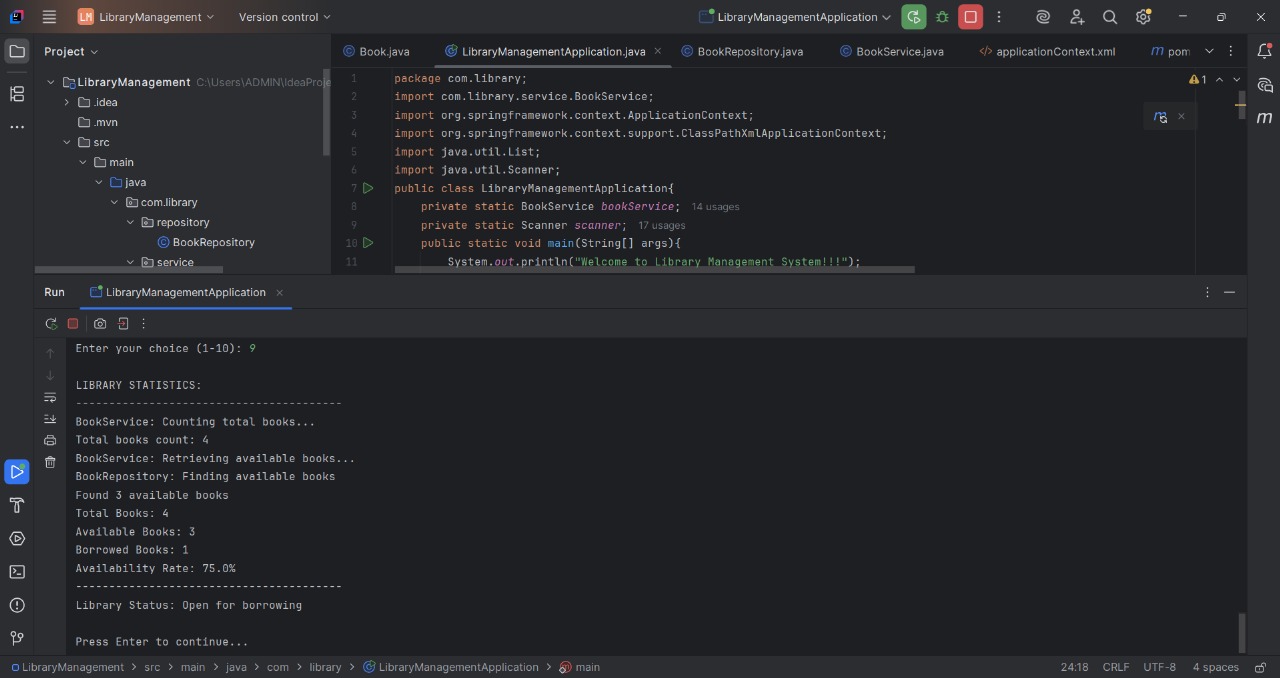
**Output:**











**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.

**Code:**

**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>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<description>Library Management Application</description>

<properties>

<maven.compiler.source>8</maven.compiler.source>

<maven.compiler.target>8</maven.compiler.target>

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

<spring.version>5.3.21</spring.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

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

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M7</version>

</plugin>

</plugins>

</build>

</project>

**LibraryApplication.java:**

package org.example;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

@Configuration

@ComponentScan

public class LibraryApplication{

public static void main(String[] args){

System.out.println("CTS-LIBRARY MANAGEMENT APPLICATION!!!");

System.out.println("Starting Spring Context...");

AnnotationConfigApplicationContext context=new AnnotationConfigApplicationContext(LibraryApplication.class);

System.out.println("Spring Context loaded successfully!");

System.out.println("Library Management Application is ready!");

System.out.println("\nMAVEN DEPENDENCIES USED...");

System.out.println("Spring Framework Dependencies:");

System.out.println("- spring-context: 5.3.21");

System.out.println("- spring-aop: 5.3.21");

System.out.println("- spring-webmvc: 5.3.21");

System.out.println("\nAdditional Dependencies:");

System.out.println("- javax.servlet-api: 4.0.1");

System.out.println("- junit: 4.13.2");

System.out.println("\nMaven Plugins:");

System.out.println("- maven-compiler-plugin: 3.8.1 (Java 1.8)");

System.out.println("\nSPRING BEANS LOADED...");

System.out.println("Available beans: " + context.getBeanDefinitionCount());

System.out.println("Bean names:");

String[] beanNames=context.getBeanDefinitionNames();

for(String beanName : beanNames){

System.out.println("- " + beanName);

}

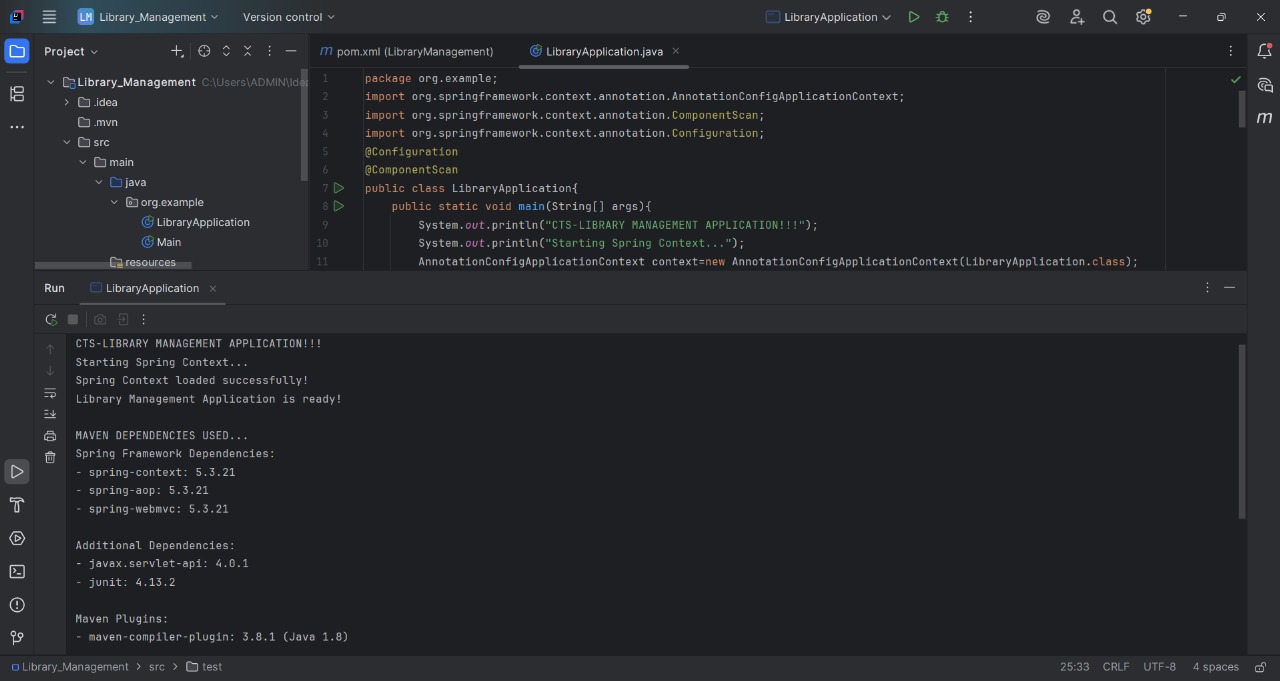
System.out.println("\nAPPLICATION READY!!!");

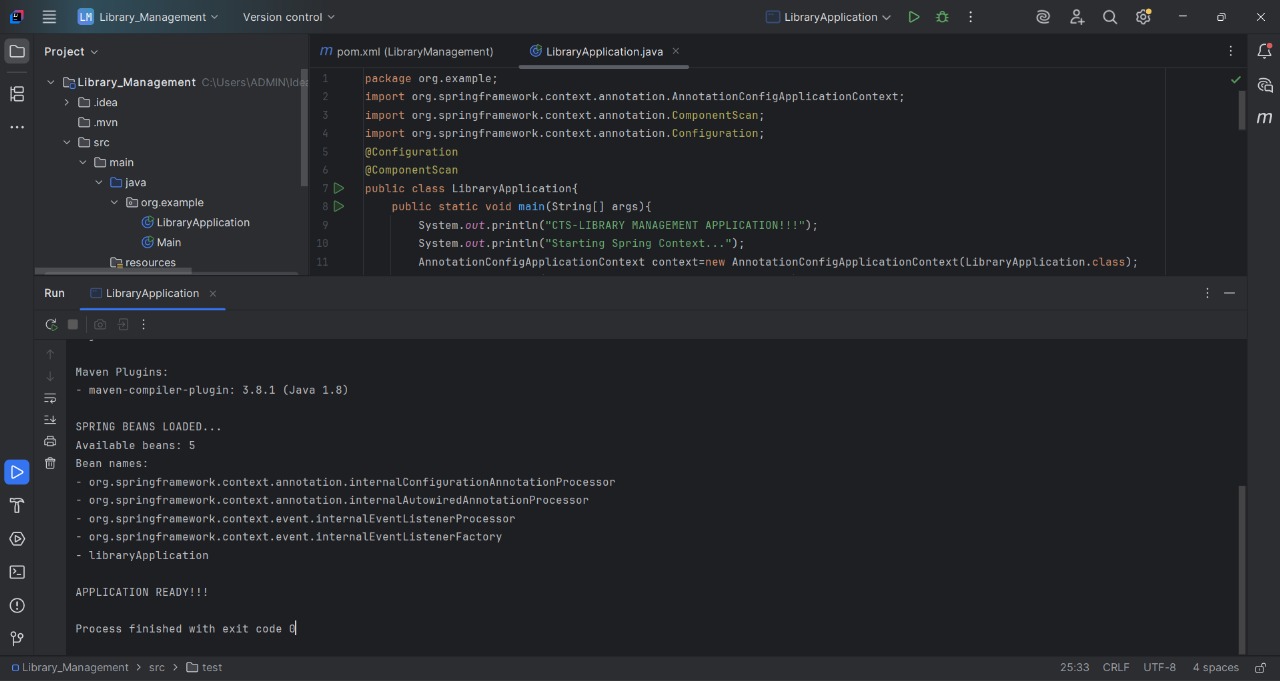
context.close();

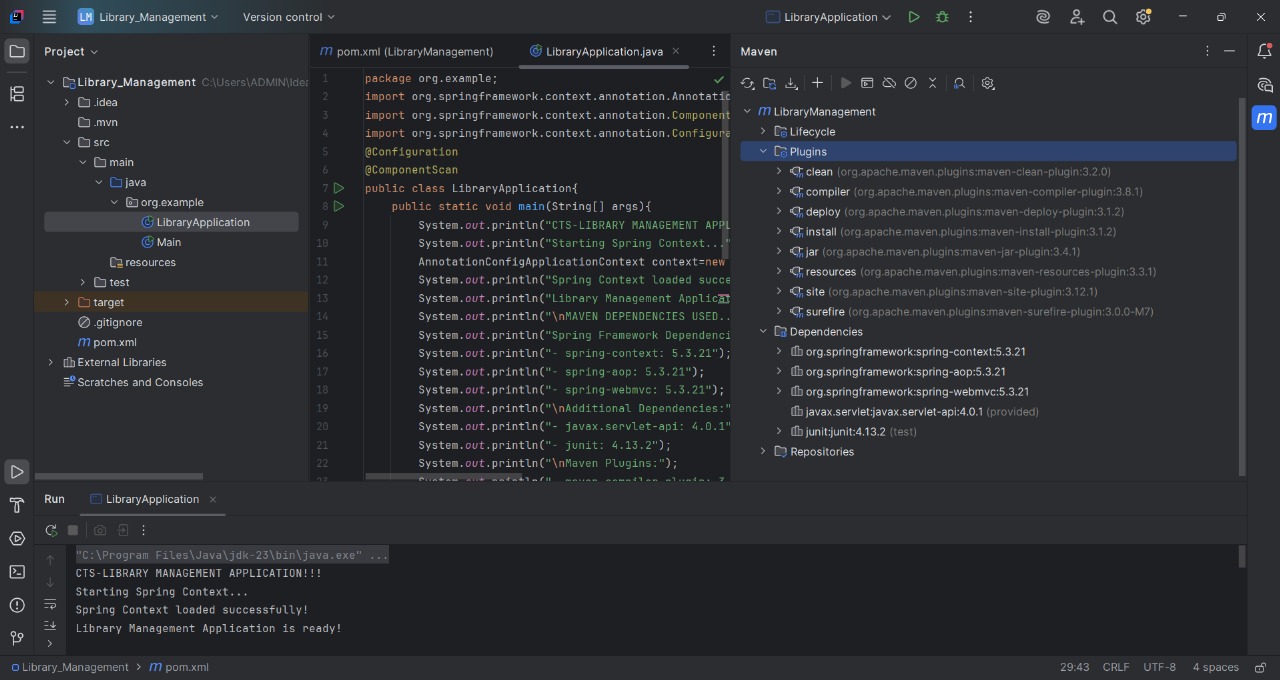
}

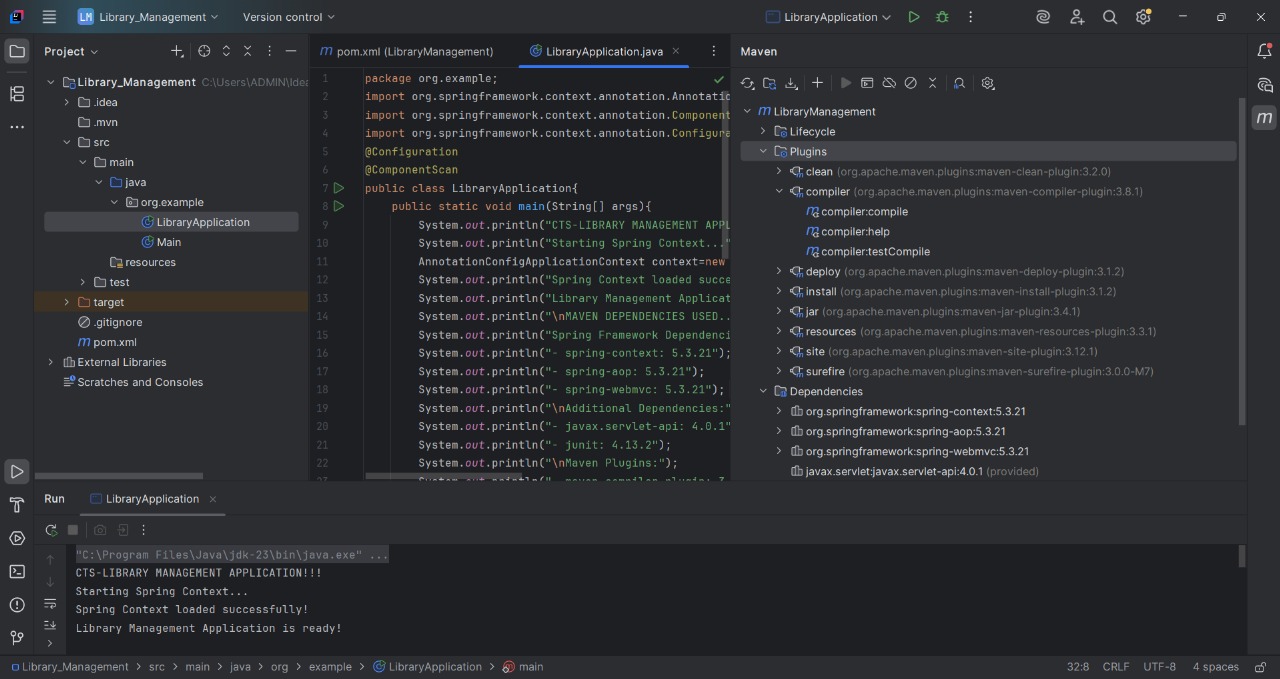
}

**Output:**









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

**Hands on 1- Spring Data JPA - Quick Example**

**Code:**

**pom.xml:**

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

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>orm-learn</name>

<description>orm-learn</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Country.java:**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Column;

import jakarta.persistence.Table;

@Entity

@Table(name="Country")

public class Country{

@Id

@Column(name="code")

private String code;

@Column(name="name")

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.ormlearn.repository;

import com.cognizant.ormlearn.model.Country;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country,String>{

}

**CountryService.java:**

package com.cognizant.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 java.util.List;

@Service

public class CountryService{

@Autowired

private CountryRepository countryRepository;

public List<Country> getAllCountries(){

return countryRepository.findAll();

}

public void saveCountry(Country country){

countryRepository.save(country);

}

}

**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.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication{

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

public static void main(String[] args){

SpringApplication.run(OrmLearnApplication.class,args);

}

@Bean

CommandLineRunner init(CountryService service){

return arguments->{

LOG.info("Application Started!!!");

List<Country> allCountries=service.getAllCountries();

if(allCountries.isEmpty()){

service.saveCountry(new Country("BR","Brazil"));

service.saveCountry(new Country("CN","China"));

service.saveCountry(new Country("ZA","South Africa"));

service.saveCountry(new Country("RU","Russia"));

service.saveCountry(new Country("NZ","New Zealand"));

LOG.info("Sample countries inserted into DB");

}

LOG.info("List of Countries...");

allCountries=service.getAllCountries();

allCountries.forEach(c->

LOG.debug("Country[code={}, name={}]",c.getCode(),c.getName())

);

LOG.info("Application Finished!!!");

};

}

}

**application.properties:**

spring.application.name=orm-learn

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

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

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

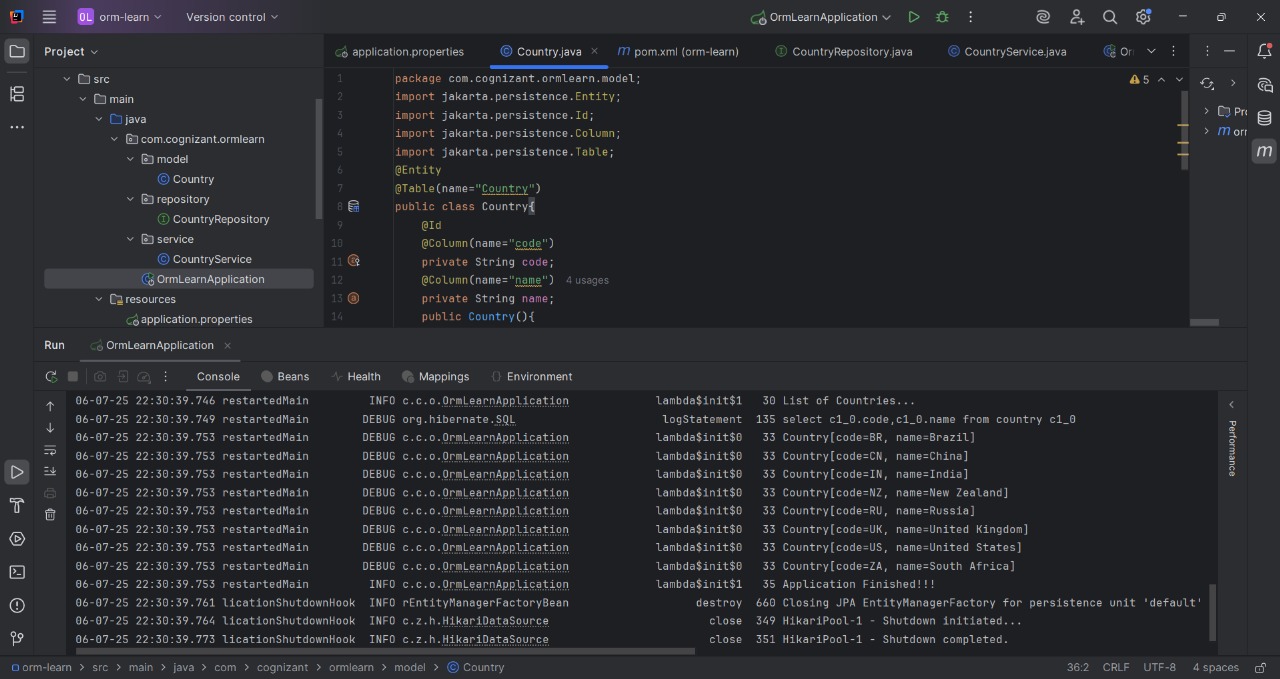
spring.datasource.username=root

spring.datasource.password=root123

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**Output:**



**Hands on 4:**

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

**Java Persistence API (JPA)**

**Definition:**

JPA (Java Persistence API) is a Java specification defined under JSR 338 that establishes a standard for Object-Relational Mapping (ORM) in Java enterprise applications.

**Key Characteristics of JPA:**

**1. Specification Nature:**

* JPA is purely a specification document, not actual code
* Defines interfaces, annotations, and contracts that implementations must follow
* Provides no executable code - only blueprints for implementation

**2. Core Components:**

* Entity Classes: POJOs marked with @Entity annotation
* EntityManager: Primary interface for persistence operations
* EntityManagerFactory: Factory for creating EntityManager instances
* Persistence Context: Environment where entity instances are managed
* JPQL (Java Persistence Query Language): Object-oriented query language

**3. Key Annotations:**

* @Entity: Marks a class as a persistent entity
* @Id: Designates primary key field
* @GeneratedValue: Specifies primary key generation strategy
* @Column: Maps entity field to database column
* @Table: Specifies database table mapping

**Hibernate :**

**Definition:**

Hibernate is a mature, feature-rich Object-Relational Mapping (ORM) framework that implements the JPA specification while providing additional proprietary features.

**Hibernate as JPA Implementation:**

**1. JPA Compliance:**

* Implements all JPA interfaces (EntityManager, EntityManagerFactory, etc.)
* Supports all JPA annotations and configurations
* Provides JPA-compliant query language (JPQL)
* Follows JPA lifecycle and persistence context rules

**2. Native Hibernate API**

* Session Interface: Hibernate's proprietary equivalent to EntityManager
* SessionFactory: Factory for creating Session instances
* Configuration: Hibernate-specific configuration mechanisms
* HQL (Hibernate Query Language): More powerful than JPQL
* Criteria API: Type-safe query construction

**Hibernate Code Analysis**

public Integer addEmployee(Employee employee){

Session session=factory.openSession();

Transaction tx=null;

Integer employeeID=null;

try{

tx=session.beginTransaction();

employeeID=(Integer) session.save(employee);

tx.commit();

}catch (HibernateException e){

if(tx!=null) tx.rollback();

e.printStackTrace();

}finally{

session.close();

}

return employeeID;

}

**Spring Data JPA:**

**Definition:**

Spring Data JPA is a higher-level abstraction framework built on top of JPA implementations that dramatically simplifies data access layer development.

**Architecture and Design:**

**1. Repository Pattern Implementation:**

* Repository Interface: Base interface for CRUD operations
* CrudRepository: Basic CRUD operations
* PagingAndSortingRepository: Adds pagination and sorting
* JpaRepository: JPA-specific operations with batch processing

**2. Proxy-Based Implementation:**

* Spring creates proxy implementations at runtime
* Uses reflection and bytecode generation
* Intercepts method calls to generate appropriate queries
* Handles transaction management transparently

**3. Transaction Management:**

* Declarative Transactions: @Transactional annotation
* Automatic Rollback: On RuntimeException and Error
* Propagation Control: REQUIRED, REQUIRES\_NEW, NESTED, etc.
* Isolation Levels: READ\_COMMITTED, REPEATABLE\_READ, etc.

**Spring Data JPA Code Analysis:**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee){

employeeRepository.save(employee);

}