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

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

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

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

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

</properties>

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

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

</dependency>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

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

</dependency>

<!-- Spring Beans -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

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

**BookRepository**

package com.library.repository;

import java.util.ArrayList;

import java.util.List;

public class BookRepository {

private List<String> books = new ArrayList<>();

public BookRepository() {

books.add("Spring in Action");

books.add("Java: The Complete Reference");

books.add("Clean Code");

System.*out*.println("BookRepository created and initialized with sample books");

}

public List<String> getAllBooks() {

return books;

}

public void addBook(String book) {

books.add(book);

System.*out*.println("Book added to repository: " + book);

}

public void displayBooks() {

System.*out*.println("=== Books in Repository ===");

for (int i = 0; i < books.size(); i++) {

System.*out*.println((i + 1) + ". " + books.get(i));

}

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

System.*out*.println("============================");

}

}

**Book Service**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public BookService() {

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

}

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

System.*out*.println("BookService created with constructor injection");

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

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

}

public void displayAllBooks() {

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

if (bookRepository != null) {

bookRepository.displayBooks();

} else {

System.*out*.println("BookRepository is not initialized!");

}

}

public void addBook(String book) {

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

if (bookRepository != null) {

bookRepository.addBook(book);

} else {

System.*out*.println("BookRepository is not initialized!");

}

}

}

**application.xml**

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

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

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

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

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

<!-- Define BookRepository bean -->

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

</bean>

<!-- Define BookService bean with dependency injection -->

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

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

</bean>

</beans>

**LibraryManagementApplication**

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {

public static void main(String[] args) {

System.*out*.println("=== Starting Library Management Application ===");

try {

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

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

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

System.*out*.println("\n=== Testing Basic Operations ===");

bookService.displayAllBooks();

bookService.addBook("Spring Boot in Action");

bookService.displayAllBooks();

System.*out*.println("\n=== Application completed successfully! ===");

} catch (Exception e) {

System.*err*.println("Error occurred: " + e.getMessage());

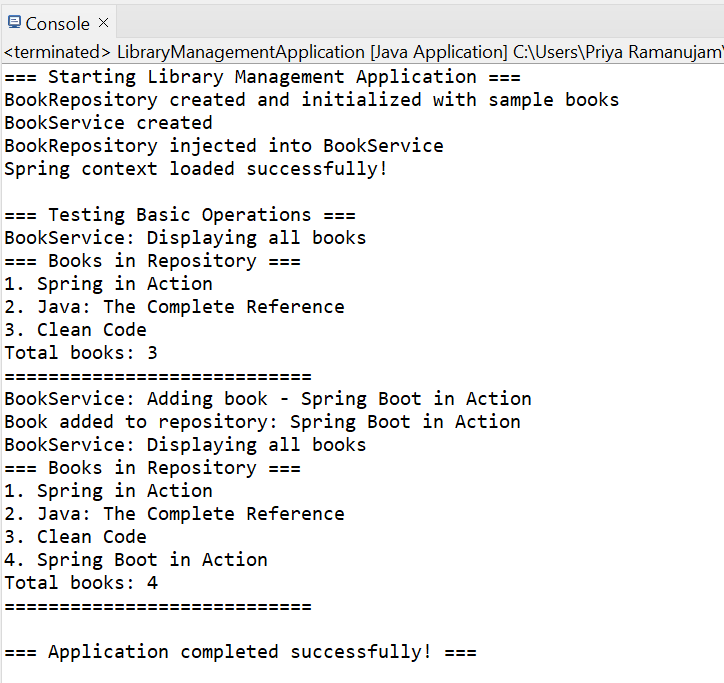
e.printStackTrace();

}

}

}

**Output:**



**Exercise 2: Implementing Dependency Injection**

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

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

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

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

</properties>

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

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

</dependency>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

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

</dependency>

<!-- Spring Beans -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.21</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.3.21</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>5.3.21</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>

**application.xml**

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

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

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

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

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

<!-- Define BookRepository bean -->

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

<!-- Define BookService bean with dependency injection -->

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

<!-- Setter-based dependency injection -->

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

</bean>

</beans>

**Book class**

package com.library.model;

public class Book {

private String id;

private String title;

private String author;

private String isbn;

public Book() {}

public Book(String id, String title, String author, String isbn) {

this.id = id;

this.title = title;

this.author = author;

this.isbn = isbn;

}

public String getId() { return id; }

public void setId(String 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; }

@Override

public String toString() {

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

}

}

**BookRepository**

package com.library.repository;

import com.library.model.Book;

import java.util.\*;

public class BookRepository {

private List<Book> books;

public BookRepository() {

this.books = new ArrayList<>();

books.add(new Book("1", "Spring in Action", "Craig Walls", "978-1617294945"));

books.add(new Book("2", "Java: The Complete Reference", "Herbert Schildt", "978-1260440232"));

books.add(new Book("3", "Clean Code", "Robert C. Martin", "978-0132350884"));

System.*out*.println("BookRepository initialized with sample data");

}

public List<Book> findAll() {

return new ArrayList<>(books);

}

public Book findById(String id) {

return books.stream()

.filter(book -> book.getId().equals(id))

.findFirst()

.orElse(null);

}

public void save(Book book) {

books.add(book);

System.*out*.println("Book saved: " + book.getTitle());

}

public void deleteById(String id) {

books.removeIf(book -> book.getId().equals(id));

System.*out*.println("Book deleted with ID: " + id);

} }

**BookService**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

import java.util.List;

public class BookService {

private BookRepository bookRepository;

public BookService() {

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

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

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

}

public List<Book> getAllBooks() {

System.*out*.println("Fetching all books...");

return bookRepository.findAll();

}

public Book getBookById(String id) {

System.*out*.println("Fetching book with ID: " + id);

return bookRepository.findById(id);

}

public void addBook(Book book) {

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

bookRepository.save(book);

}

public void removeBook(String id) {

System.*out*.println("Removing book with ID: " + id);

bookRepository.deleteById(id);

}

public void displayAllBooks() {

List<Book> books = getAllBooks();

System.*out*.println("\n=== Library Books ===");

if (books.isEmpty()) {

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

} else {

books.forEach(System.*out*::println);

}

System.*out*.println("=====================\n");

}

}

**LibraryManagementApplication**

package com.library.main;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

import com.library.model.Book;

public class LibraryManagementApplication {

public static void main(String[] args) {

System.*out*.println("Starting Library Management Application...");

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

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

System.*out*.println("\n=== Testing Dependency Injection ===");

bookService.displayAllBooks();

Book newBook = new Book("4", "Spring Boot in Action", "Craig Walls", "978-1617292545");

bookService.addBook(newBook);

bookService.displayAllBooks();

Book foundBook = bookService.getBookById("1");

if (foundBook != null) {

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

}

bookService.removeBook("2");

bookService.displayAllBooks();

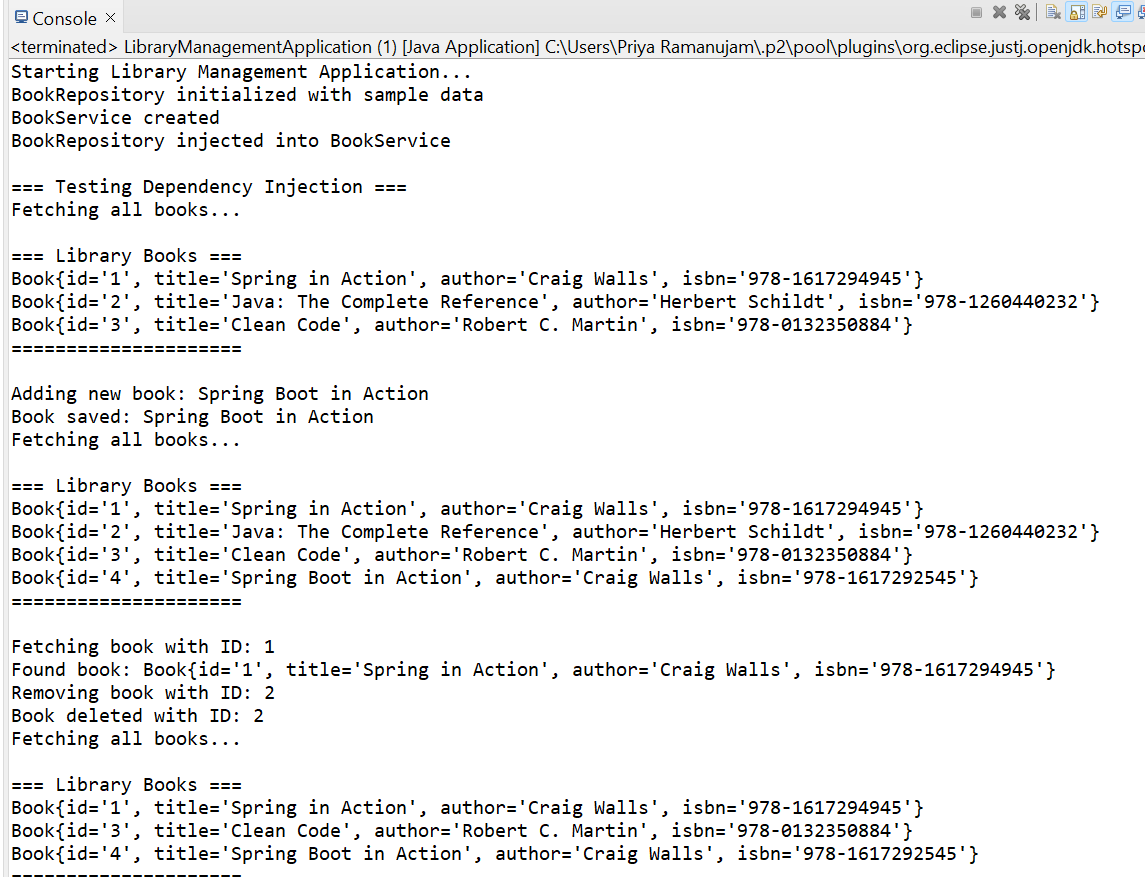
System.*out*.println("Application completed successfully!");

((ClassPathXmlApplicationContext) context).close();

}

}

**Output:**



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

**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>

<!-- Project Information -->

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>Library Management System</name>

<description>A Spring-based Library Management Application</description>

<!-- Properties -->

<properties>

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

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

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

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

</properties>

<!-- Dependencies -->

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

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

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

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

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

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

</dependency>

<!-- AspectJ Weaver (for AOP) -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.7</version>

</dependency>

<!-- Servlet API (for Web MVC) -->

<dependency>

<groupId>javax.servlet</groupId>

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

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

<!-- JSTL (for JSP views) -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- JUnit for testing -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

<!-- Spring Test -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

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

<scope>test</scope>

</dependency>

</dependencies>

<!-- Build Configuration -->

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<plugin>

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

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

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

<!-- Maven Surefire Plugin (for running tests) -->

<plugin>

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

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

<version>3.0.0-M7</version>

</plugin>

<!-- Maven War Plugin (if you want to create WAR files) -->

<plugin>

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

<artifactId>maven-war-plugin</artifactId>

<version>3.2.3</version>

<configuration>

<warSourceDirectory>src/main/webapp</warSourceDirectory>

</configuration>

</plugin>

</plugins>

</build>

</project>

**applicationContent.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"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xsi:schemaLocation="

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

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

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

http://www.springframework.org/schema/context/spring-context.xsd

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

http://www.springframework.org/schema/aop/spring-aop.xsd

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

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

<!-- Enable component scanning -->

<context:component-scan base-package="com.library" />

<!-- Enable AOP -->

<aop:aspectj-autoproxy />

<!-- Enable Spring MVC -->

<mvc:annotation-driven />

<!-- Configure View Resolver -->

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/views/" />

<property name="suffix" value=".jsp" />

</bean>

<!-- Book Repository Bean -->

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

<!-- Book Service Bean -->

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

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

</bean>

</beans>

**Book class**

package com.library.model;

public class Book {

private String id;

private String title;

private String author;

private String isbn;

public Book() {}

public Book(String id, String title, String author, String isbn) {

this.id = id;

this.title = title;

this.author = author;

this.isbn = isbn;

}

public String getId() { return id; }

public void setId(String 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; }

@Override

public String toString() {

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

}

}

**BookRepository**

package com.library.repository;

import com.library.model.Book;

import java.util.\*;

public class BookRepository {

private List<Book> books;

public BookRepository() {

this.books = new ArrayList<>();

books.add(new Book("1", "Spring in Action", "Craig Walls", "978-1617294945"));

books.add(new Book("2", "Java: The Complete Reference", "Herbert Schildt", "978-1260440232"));

books.add(new Book("3", "Clean Code", "Robert C. Martin", "978-0132350884"));

System.*out*.println("BookRepository initialized with sample data");

}

public List<Book> findAll() {

return new ArrayList<>(books);

}

public Book findById(String id) {

return books.stream()

.filter(book -> book.getId().equals(id))

.findFirst()

.orElse(null);

}

public void save(Book book) {

books.add(book);

System.*out*.println("Book saved: " + book.getTitle());

}

public void deleteById(String id) {

books.removeIf(book -> book.getId().equals(id));

System.*out*.println("Book deleted with ID: " + id);

} }

**BookService**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

import java.util.List;

public class BookService {

private BookRepository bookRepository;

public BookService() {

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

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

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

}

public List<Book> getAllBooks() {

System.*out*.println("Fetching all books...");

return bookRepository.findAll();

}

public Book getBookById(String id) {

System.*out*.println("Fetching book with ID: " + id);

return bookRepository.findById(id);

}

public void addBook(Book book) {

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

bookRepository.save(book);

}

public void removeBook(String id) {

System.*out*.println("Removing book with ID: " + id);

bookRepository.deleteById(id);

}

public void displayAllBooks() {

List<Book> books = getAllBooks();

System.*out*.println("\n=== Library Books ===");

if (books.isEmpty()) {

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

} else {

books.forEach(System.*out*::println);

}

System.*out*.println("=====================\n");

}

}

**LibraryManagementApplication**

package com.library.main;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

import com.library.model.Book;

public class LibraryManagementApplication {

public static void main(String[] args) {

System.*out*.println("Starting Library Management Application...");

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

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

System.*out*.println("\n=== Testing Dependency Injection ===");

bookService.displayAllBooks();

Book newBook = new Book("4", "Spring Boot in Action", "Craig Walls", "978-1617292545");

bookService.addBook(newBook);

bookService.displayAllBooks();

Book foundBook = bookService.getBookById("1");

if (foundBook != null) {

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

}

bookService.removeBook("2");

bookService.displayAllBooks();

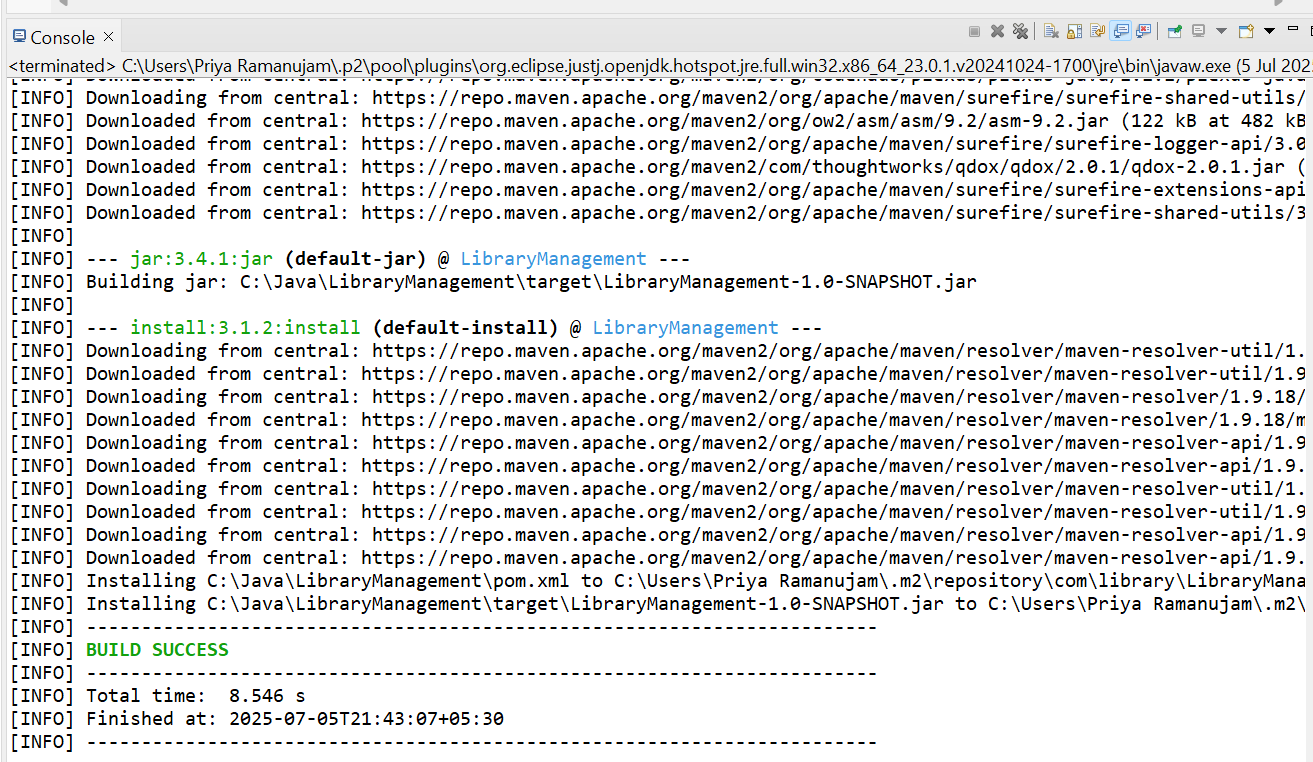
System.*out*.println("Application completed successfully!");

((ClassPathXmlApplicationContext) context).close();

}

}

**Output:**



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

**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>library-management</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>Library Management System</name>

<description>A Spring-based Library Management Application</description>

<properties>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

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

<spring.version>5.3.23</spring.version>

</properties>

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

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

</dependency>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

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

</dependency>

<!-- Spring Beans -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

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

</dependency>

<!-- JUnit for testing -->

<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>11</source>

<target>11</target>

</configuration>

</plugin>

<plugin>

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

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

<version>3.0.0</version>

<configuration>

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

</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">

<!-- Define BookRepository bean -->

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

<!-- Define BookService bean with dependency injection -->

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

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

</bean>

</beans>

**BookService class**

package com.library.service;

import com.library.repository.BookRepository;

import com.library.model.Book;

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;

}

public void addBook(Book book) {

if (book != null && book.getTitle() != null && !book.getTitle().trim().isEmpty()) {

bookRepository.save(book);

System.*out*.println("Book added: " + book.getTitle());

} else {

System.*out*.println("Invalid book data");

}

}

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

public Book findBookById(Long id) {

return bookRepository.findById(id);

}

public void updateBook(Book book) {

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

bookRepository.update(book);

System.*out*.println("Book updated: " + book.getTitle());

} else {

System.*out*.println("Invalid book data for update");

}

}

public void deleteBook(Long id) {

if (id != null) {

bookRepository.delete(id);

System.*out*.println("Book deleted with ID: " + id);

}

}

public void displayAllBooks() {

List<Book> books = getAllBooks();

if (books.isEmpty()) {

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

} else {

System.*out*.println("Library Books:");

for (Book book : books) {

System.*out*.println("- " + book.getTitle() + " by " + book.getAuthor());

}

}

}

}

**BookRepository Class**

package com.library.repository;

import com.library.model.Book;

import java.util.ArrayList;

import java.util.List;

public class BookRepository {

private List<Book> books;

public BookRepository() {

this.books = new ArrayList<>();

initializeSampleData();

}

private void initializeSampleData() {

books.add(new Book(1L, "Java: The Complete Reference", "Herbert Schildt", "978-1260440232"));

books.add(new Book(2L, "Spring in Action", "Craig Walls", "978-1617294945"));

books.add(new Book(3L, "Clean Code", "Robert C. Martin", "978-0132350884"));

}

public void save(Book book) {

if (book != null) {

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

book.setId(generateNextId());

}

books.add(book);

}

}

public List<Book> findAll() {

return new ArrayList<>(books);

}

public Book findById(Long id) {

if (id == null) {

return null;

}

return books.stream()

.filter(book -> id.equals(book.getId()))

.findFirst()

.orElse(null);

}

public void update(Book updatedBook) {

if (updatedBook == null || updatedBook.getId() == null) {

return;

}

for (int i = 0; i < books.size(); i++) {

Book book = books.get(i);

if (updatedBook.getId().equals(book.getId())) {

books.set(i, updatedBook);

break;

}

}

}

public void delete(Long id) {

if (id == null) {

return;

}

books.removeIf(book -> id.equals(book.getId()));

}

public List<Book> findByAuthor(String author) {

if (author == null) {

return new ArrayList<>();

}

return books.stream()

.filter(book -> author.equalsIgnoreCase(book.getAuthor()))

.toList();

}

public List<Book> findByTitle(String title) {

if (title == null) {

return new ArrayList<>();

}

return books.stream()

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

.toList();

}

private Long generateNextId() {

return books.stream()

.mapToLong(Book::getId)

.max()

.orElse(0L) + 1;

}

public int count() {

return books.size();

}

}

**Book.java**

package com.library.model;

public class Book {

private Long id;

private String title;

private String author;

private String isbn;

public Book() {

}

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

this.id = id;

this.title = title;

this.author = author;

this.isbn = isbn;

}

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

this.title = title;

this.author = author;

this.isbn = isbn;

}

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;

}

@Override

public String toString() {

return "Book{" +

"id=" + id +

", title='" + title + '\'' +

", author='" + author + '\'' +

", isbn='" + isbn + '\'' +

'}';

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Book book = (Book) obj;

return id != null && id.equals(book.id);

}

@Override

public int hashCode() {

return id != null ? id.hashCode() : 0;

}

}

**LibraryManagementApplication**

package com.library.main;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

import com.library.model.Book;

public class LibraryManagementApplication {

public static void main(String[] args) {

System.*out*.println("Starting Library Management Application...");

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

try {

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

System.*out*.println("Spring IoC Container configured successfully!");

*testLibraryOperations*(bookService);

} catch (Exception e) {

System.*err*.println("Error: " + e.getMessage());

e.printStackTrace();

} finally {

if (context instanceof ClassPathXmlApplicationContext) {

((ClassPathXmlApplicationContext) context).close();

}

}

}

private static void testLibraryOperations(BookService bookService) {

System.*out*.println("\n=== Testing Library Operations ===");

Book book1 = new Book(1L, "The Great Gatsby", "F. Scott Fitzgerald", "978-0-7432-7356-5");

Book book2 = new Book(2L, "To Kill a Mockingbird", "Harper Lee", "978-0-06-112008-4");

Book book3 = new Book(3L, "1984", "George Orwell", "978-0-452-28423-4");

bookService.addBook(book1);

bookService.addBook(book2);

bookService.addBook(book3);

System.*out*.println("\n=== All Books in Library ===");

bookService.displayAllBooks();

System.*out*.println("\n=== Finding Book by ID ===");

Book foundBook = bookService.findBookById(2L);

if (foundBook != null) {

System.*out*.println("Found: " + foundBook.getTitle() + " by " + foundBook.getAuthor());

}

System.*out*.println("\n=== Updating Book ===");

book1.setTitle("The Great Gatsby - Updated Edition");

bookService.updateBook(book1);

System.*out*.println("\n=== Books After Update ===");

bookService.displayAllBooks();

System.*out*.println("\n=== Deleting Book ===");

bookService.deleteBook(3L);

System.*out*.println("\n=== Final Book List ===");

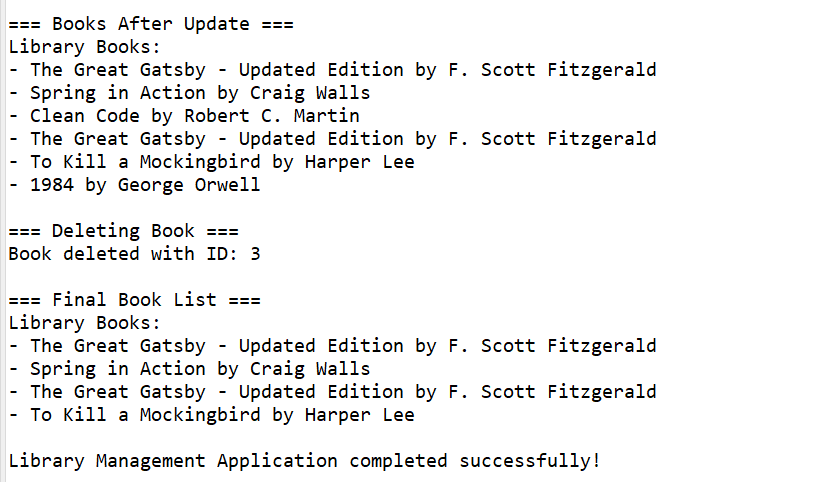
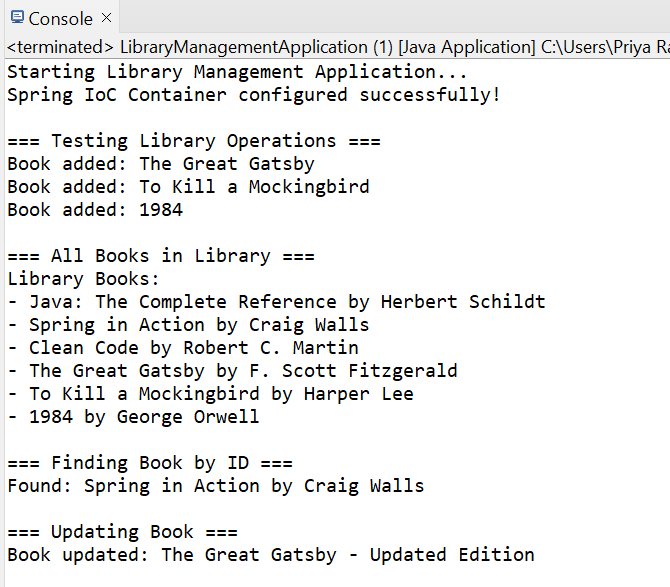
bookService.displayAllBooks();

System.*out*.println("\nLibrary Management Application completed successfully!");

}

}

**Output:**

****

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

**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>library-management</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>Library Management System</name>

<description>A Spring-based Library Management Application</description>

<properties>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

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

<spring.version>5.3.23</spring.version>

</properties>

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

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

</dependency>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

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

</dependency>

<!-- Spring Beans -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

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

</dependency>

<!-- JUnit for testing -->

<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>11</source>

<target>11</target>

</configuration>

</plugin>

<plugin>

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

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

<version>3.0.0</version>

<configuration>

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

</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">

<!-- BookRepository Bean -->

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

<constructor-arg name="initialCapacity" value="10"/>

</bean>

<!-- BookService with Constructor Injection -->

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

<constructor-arg ref="bookRepository"/>

</bean>

<!-- BookService with Setter Injection -->

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

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

</bean>

<!-- Alternative: BookService with both Constructor and Setter Injection -->

<!-- This demonstrates that constructor injection happens first, then setter injection -->

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

<constructor-arg ref="bookRepository"/>

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

</bean>

</beans>

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

import com.library.model.Book;

import java.util.List;

public class BookService {

private BookRepository bookRepository;

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

System.*out*.println("BookService created using Constructor Injection");

}

public BookService() {

System.*out*.println("BookService created using Default Constructor");

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

System.*out*.println("BookRepository injected using Setter Injection");

}

public void addBook(Book book) {

bookRepository.save(book);

System.*out*.println("Book added: " + book.getTitle());

}

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

public Book getBookById(Long id) {

return bookRepository.findById(id);

}

public void displayAllBooks() {

List<Book> books = getAllBooks();

System.*out*.println("\n=== Library Books ===");

if (books.isEmpty()) {

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

} else {

books.forEach(book ->

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

", Title: " + book.getTitle() +

", Author: " + book.getAuthor())

);

}

} }

**BookRepository.java**

package com.library.repository;

import com.library.model.Book;

import java.util.ArrayList;

import java.util.List;

public class BookRepository {

private List<Book> books;

public BookRepository(int initialCapacity) {

this.books = new ArrayList<>(initialCapacity);

System.*out*.println("BookRepository created with initial capacity: " + initialCapacity);

}

public BookRepository() {

this.books = new ArrayList<>();

System.*out*.println("BookRepository created with default constructor");

}

public void save(Book book) {

books.add(book);

System.*out*.println("Book saved in repository: " + book.getTitle());

}

public List<Book> findAll() {

return new ArrayList<>(books);

}

public Book findById(Long id) {

return books.stream()

.filter(book -> book.getId().equals(id))

.findFirst()

.orElse(null);

}

public void deleteById(Long id) {

books.removeIf(book -> book.getId().equals(id));

}

public int count() {

return books.size();

}

}

**Book.java**

package com.library.model;

public class Book {

private Long id;

private String title;

private String author;

public Book() {

}

public Book(Long id, String title, String author) {

this.id = id;

this.title = title;

this.author = author;

}

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;

}

@Override

public String toString() {

return "Book{" +

"id=" + id +

", title='" + title + '\'' +

", author='" + author + '\'' +

'}';

}

}

**LibraryManagementApplication**

package com.library.main;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

import com.library.model.Book;

public class LibraryManagementApplication {

public static void main(String[] args) {

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

System.*out*.println("=== Spring Context Loaded ===\n");

System.*out*.println("1. Testing Constructor Injection:");

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

*testBookService*(bookServiceConstructor, "Constructor Injection");

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

System.*out*.println("2. Testing Setter Injection:");

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

*testBookService*(bookServiceSetter, "Setter Injection");

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

System.*out*.println("3. Testing Both Constructor and Setter Injection:");

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

*testBookService*(bookServiceBoth, "Both Injection Types");

((ClassPathXmlApplicationContext) context).close();

}

private static void testBookService(BookService bookService, String injectionType) {

System.*out*.println("Testing " + injectionType + ":");

bookService.addBook(new Book(1L, "Spring in Action", "Craig Walls"));

bookService.addBook(new Book(2L, "Java: The Complete Reference", "Herbert Schildt"));

bookService.addBook(new Book(3L, "Clean Code", "Robert C. Martin"));

bookService.displayAllBooks();

Book book = bookService.getBookById(1L);

if (book != null) {

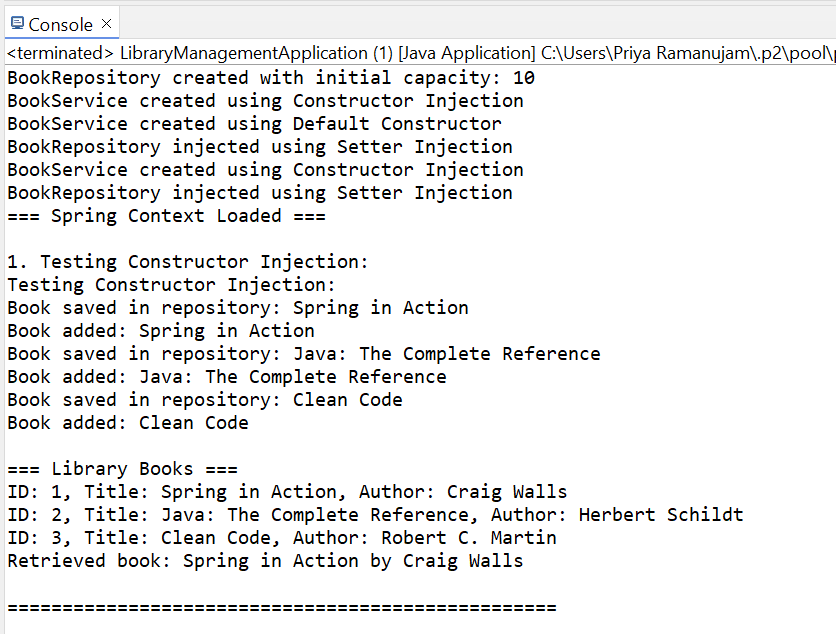
System.*out*.println("Retrieved book: " + book.getTitle() + " by " + book.getAuthor());

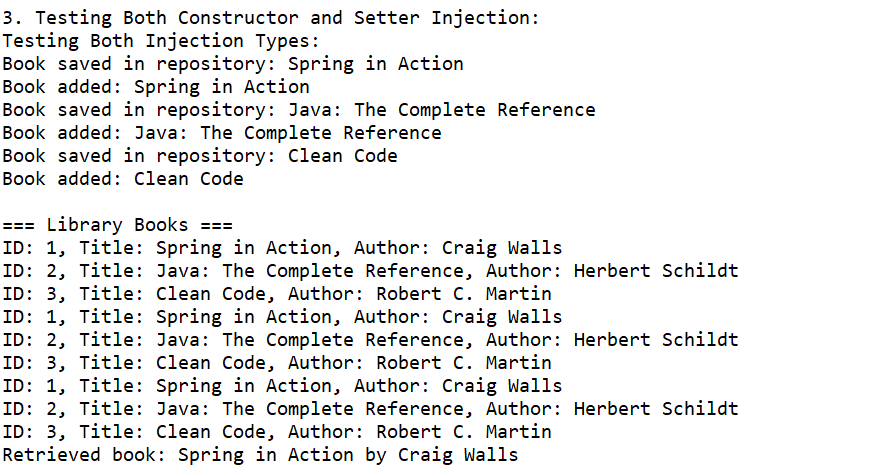
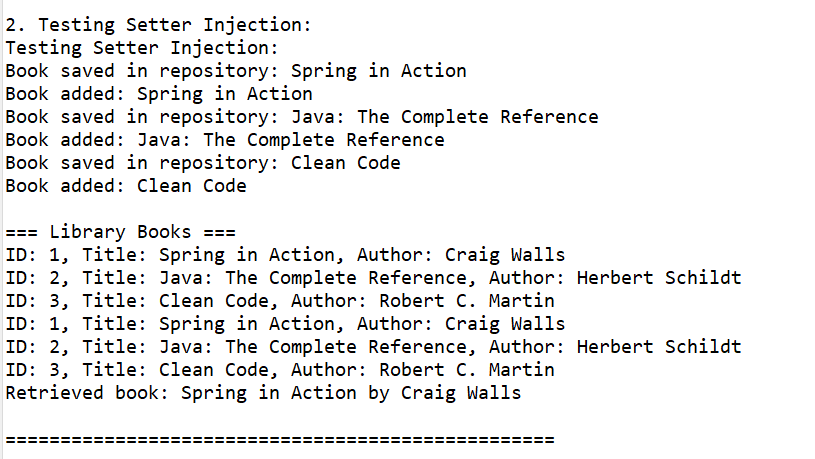
}

}

}

**Output:**

****

****

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

**pom.xml**

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

<project xmlns="http://maven.apache.org/POM/4.0.0"

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

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

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>2.7.14</version>

<relativePath/>

</parent>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

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

<properties>

<java.version>1.8</java.version>

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

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

</properties>

<dependencies>

<!-- Spring Boot Web Starter -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Data JPA Starter -->

<dependency>

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

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

</dependency>

<!-- H2 Database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Test Starter -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<!-- Spring Boot Validation -->

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**application.properties**

# Server Configuration

server.port=8080

server.servlet.context-path=/api

# H2 Database Configuration

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

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

spring.datasource.username=sa

spring.datasource.password=password

# JPA/Hibernate Configuration

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

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true

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

# H2 Console (for testing purposes)

spring.h2.console.enabled=true

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

# Logging Configuration

logging.level.com.library=DEBUG

logging.level.org.springframework.web=DEBUG

logging.level.org.hibernate.SQL=DEBUG

**BookService**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class BookService {

private final BookRepository bookRepository;

@Autowired

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public Book createBook(Book book) {

return bookRepository.save(book);

}

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

public Optional<Book> getBookById(Long id) {

return bookRepository.findById(id);

}

public Book updateBook(Long id, Book bookDetails) {

Optional<Book> optionalBook = bookRepository.findById(id);

if (optionalBook.isPresent()) {

Book book = optionalBook.get();

book.setTitle(bookDetails.getTitle());

book.setAuthor(bookDetails.getAuthor());

book.setIsbn(bookDetails.getIsbn());

book.setPublicationYear(bookDetails.getPublicationYear());

book.setAvailable(bookDetails.getAvailable());

return bookRepository.save(book);

}

return null;

}

public boolean deleteBook(Long id) {

if (bookRepository.existsById(id)) {

bookRepository.deleteById(id);

return true;

}

return false;

}

public List<Book> searchByTitle(String title) {

return bookRepository.findByTitleContainingIgnoreCase(title);

}

public List<Book> searchByAuthor(String author) {

return bookRepository.findByAuthorContainingIgnoreCase(author);

}

public List<Book> getAvailableBooks() {

return bookRepository.findByAvailable(true);

}

public List<Book> getBooksByYear(Integer year) {

return bookRepository.findByPublicationYear(year);

}

public Book findByIsbn(String isbn) {

return bookRepository.findByIsbn(isbn);

}

public long getTotalBooks() {

return bookRepository.count();

}

public long getAvailableBookCount() {

return bookRepository.countAvailableBooks();

}

}

**BookRepository**

package com.library.repository;

import com.library.model.Book;

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

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

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface BookRepository extends JpaRepository<Book, Long> {

List<Book> findByTitleContainingIgnoreCase(String title);

List<Book> findByAuthorContainingIgnoreCase(String author);

List<Book> findByAvailable(Boolean available);

List<Book> findByPublicationYear(Integer year);

Book findByIsbn(String isbn);

@Query("SELECT b FROM Book b WHERE b.title LIKE %:title% AND b.author LIKE %:author%")

List<Book> findByTitleAndAuthor(@Param("title") String title, @Param("author") String author);

@Query("SELECT COUNT(b) FROM Book b WHERE b.available = true")

Long countAvailableBooks();

List<Book> findByPublicationYearGreaterThan(Integer year);

}

**Book.java**

package com.library.model;

import javax.persistence.\*;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

@Entity

@Table(name = "books")

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

private Long id;

@NotBlank(message = "Title is required")

@Size(min = 1, max = 255, message = "Title must be between 1 and 255 characters")

@Column(name = "title", nullable = false)

private String title;

@NotBlank(message = "Author is required")

@Size(min = 1, max = 255, message = "Author must be between 1 and 255 characters")

@Column(name = "author", nullable = false)

private String author;

@Column(name = "isbn")

private String isbn;

@Column(name = "publication\_year")

private Integer publicationYear;

@Column(name = "available", nullable = false)

private Boolean available = true;

public Book() {

}

public Book(String title, String author) {

this.title = title;

this.author = author;

}

public Book(String title, String author, String isbn, Integer publicationYear) {

this.title = title;

this.author = author;

this.isbn = isbn;

this.publicationYear = publicationYear;

}

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 Integer getPublicationYear() {

return publicationYear;

}

public void setPublicationYear(Integer publicationYear) {

this.publicationYear = publicationYear;

}

public Boolean getAvailable() {

return available;

}

public void setAvailable(Boolean available) {

this.available = available;

}

@Override

public String toString() {

return "Book{" +

"id=" + id +

", title='" + title + '\'' +

", author='" + author + '\'' +

", isbn='" + isbn + '\'' +

", publicationYear=" + publicationYear +

", available=" + available +

'}';

}

}

**LibraryManagementApplication**

package com.library.main;

import com.library.model.Book;

import com.library.service.BookService;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@SpringBootApplication

@ComponentScan(basePackages = "com.library")

@EntityScan(basePackages = "com.library.model")

@EnableJpaRepositories(basePackages = "com.library.repository")

public class LibraryManagementApplication implements CommandLineRunner {

@Autowired

private BookService bookService;

public static void main(String[] args) {

SpringApplication.*run*(LibraryManagementApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

System.*out*.println("=== Library Management System Started ===");

insertSampleBooks();

System.*out*.println("\n=== Sample Books Added ===");

bookService.getAllBooks().forEach(System.*out*::println);

System.*out*.println("\n=== REST API Endpoints Available ===");

System.*out*.println("GET /api/books - Get all books");

System.*out*.println("GET /api/books/{id} - Get book by ID");

System.*out*.println("POST /api/books - Create new book");

System.*out*.println("PUT /api/books/{id} - Update book");

System.*out*.println("DELETE /api/books/{id} - Delete book");

System.*out*.println("GET /api/books/search/title?q={title} - Search by title");

System.*out*.println("GET /api/books/search/author?q={author} - Search by author");

System.*out*.println("GET /api/books/available - Get available books");

System.*out*.println("GET /api/books/year/{year} - Get books by year");

System.*out*.println("GET /api/books/isbn/{isbn} - Find book by ISBN");

System.*out*.println("GET /api/books/stats - Get library statistics");

System.*out*.println("\n=== H2 Console Available ===");

System.*out*.println("URL: http://localhost:8080/api/h2-console");

System.*out*.println("JDBC URL: jdbc:h2:mem:librarydb");

System.*out*.println("Username: sa");

System.*out*.println("Password: password");

System.*out*.println("\n=== Application is running on http://localhost:8080 ===");

}

private void insertSampleBooks() {

Book book1 = new Book("Spring Boot in Action", "Craig Walls", "978-1617292545", 2015);

Book book2 = new Book("Java: The Complete Reference", "Herbert Schildt", "978-1260440232", 2020);

Book book3 = new Book("Clean Code", "Robert C. Martin", "978-0132350884", 2008);

Book book4 = new Book("Design Patterns", "Gang of Four", "978-0201633612", 1994);

Book book5 = new Book("Spring in Action", "Craig Walls", "978-1617294945", 2018);

book1.setAvailable(true);

book2.setAvailable(false);

book3.setAvailable(true);

book4.setAvailable(true);

book5.setAvailable(false);

bookService.createBook(book1);

bookService.createBook(book2);

bookService.createBook(book3);

bookService.createBook(book4);

bookService.createBook(book5);

}

}

**BookController.java**

package com.library.controller;

import com.library.model.Book;

import com.library.service.BookService;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import javax.validation.Valid;

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/books")

@CrossOrigin(origins = "\*")

public class BookController {

private final BookService bookService;

@Autowired

public BookController(BookService bookService) {

this.bookService = bookService;

}

@GetMapping

public ResponseEntity<List<Book>> getAllBooks() {

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

return ResponseEntity.*ok*(books);

}

@GetMapping("/{id}")

public ResponseEntity<Book> getBookById(@PathVariable Long id) {

Optional<Book> book = bookService.getBookById(id);

return book.map(ResponseEntity::*ok*)

.orElse(ResponseEntity.*notFound*().build());

}

@PostMapping

public ResponseEntity<Book> createBook(@Valid @RequestBody Book book) {

Book createdBook = bookService.createBook(book);

return ResponseEntity.*status*(HttpStatus.*CREATED*).body(createdBook);

}

@PutMapping("/{id}")

public ResponseEntity<Book> updateBook(@PathVariable Long id,

@Valid @RequestBody Book bookDetails) {

Book updatedBook = bookService.updateBook(id, bookDetails);

return updatedBook != null ? ResponseEntity.*ok*(updatedBook)

: ResponseEntity.*notFound*().build();

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteBook(@PathVariable Long id) {

boolean deleted = bookService.deleteBook(id);

return deleted ? ResponseEntity.*noContent*().build()

: ResponseEntity.*notFound*().build();

}

@GetMapping("/search/title")

public ResponseEntity<List<Book>> searchByTitle(@RequestParam String q) {

List<Book> books = bookService.searchByTitle(q);

return ResponseEntity.*ok*(books);

}

@GetMapping("/search/author")

public ResponseEntity<List<Book>> searchByAuthor(@RequestParam String q) {

List<Book> books = bookService.searchByAuthor(q);

return ResponseEntity.*ok*(books);

}

@GetMapping("/available")

public ResponseEntity<List<Book>> getAvailableBooks() {

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

return ResponseEntity.*ok*(books);

}

@GetMapping("/year/{year}")

public ResponseEntity<List<Book>> getBooksByYear(@PathVariable Integer year) {

List<Book> books = bookService.getBooksByYear(year);

return ResponseEntity.*ok*(books);

}

@GetMapping("/isbn/{isbn}")

public ResponseEntity<Book> findByIsbn(@PathVariable String isbn) {

Book book = bookService.findByIsbn(isbn);

return book != null ? ResponseEntity.*ok*(book)

: ResponseEntity.*notFound*().build();

}

@GetMapping("/stats")

public ResponseEntity<LibraryStats> getLibraryStats() {

LibraryStats stats = new LibraryStats();

stats.setTotalBooks(bookService.getTotalBooks());

stats.setAvailableBooks(bookService.getAvailableBookCount());

stats.setBorrowedBooks(stats.getTotalBooks() - stats.getAvailableBooks());

return ResponseEntity.*ok*(stats);

}

public static class LibraryStats {

private long totalBooks;

private long availableBooks;

private long borrowedBooks;

public long getTotalBooks() { return totalBooks; }

public void setTotalBooks(long totalBooks) { this.totalBooks = totalBooks; }

public long getAvailableBooks() { return availableBooks; }

public void setAvailableBooks(long availableBooks) { this.availableBooks = availableBooks; }

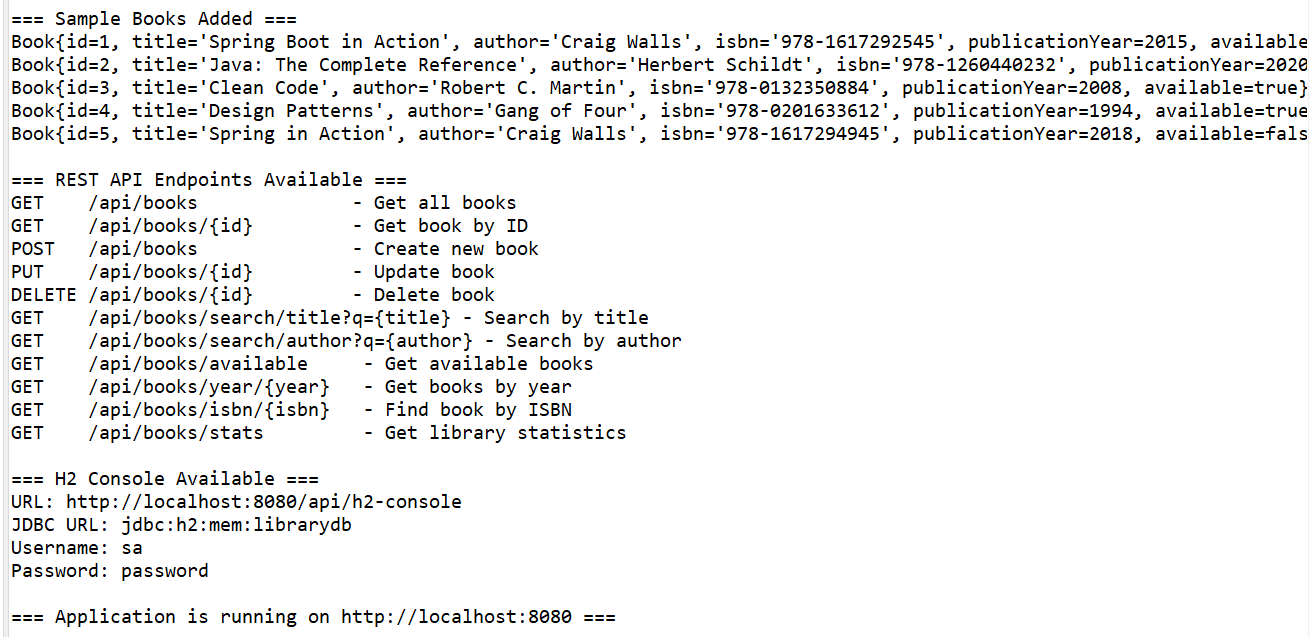
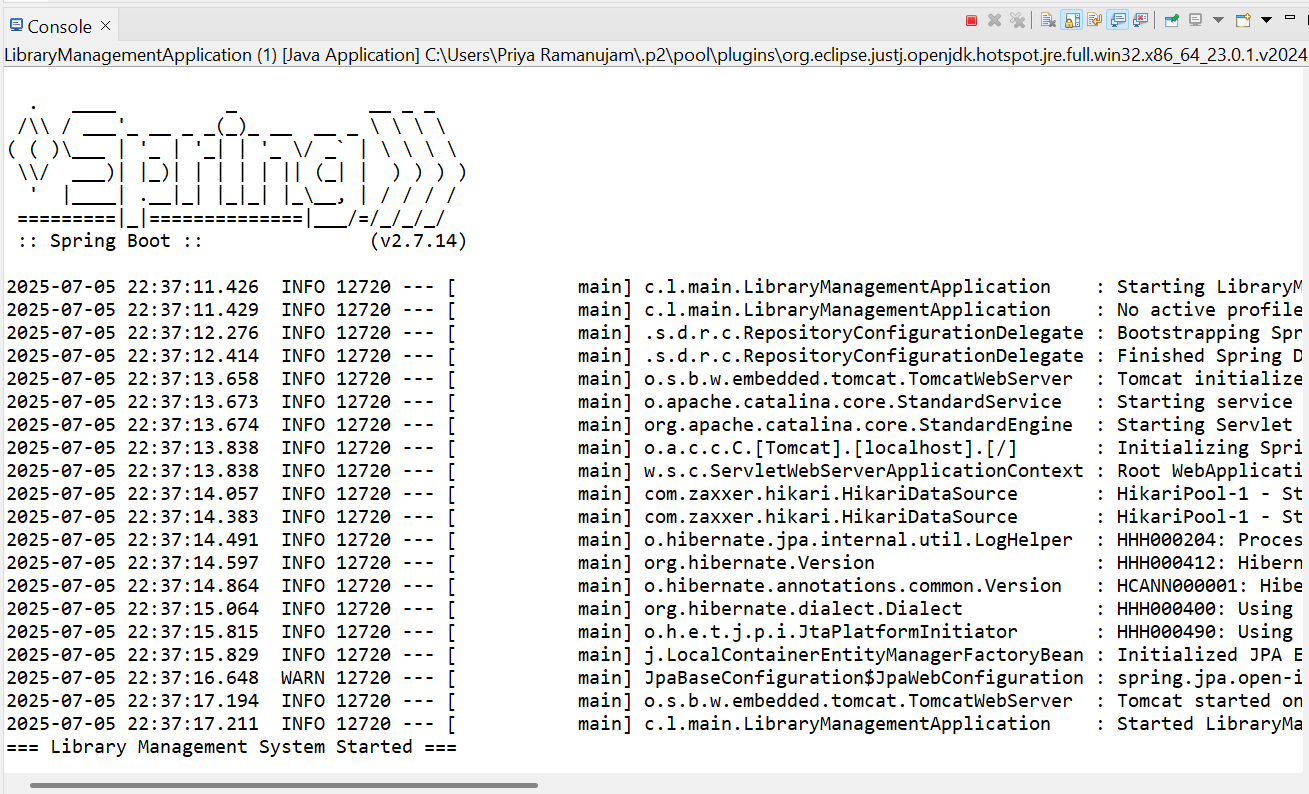
public long getBorrowedBooks() { return borrowedBooks; }

public void setBorrowedBooks(long borrowedBooks) { this.borrowedBooks = borrowedBooks; }

}

}

**Output:**

****