**WEEK -3**

**Spring Core And Maven**

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

**Package name: com.exercise1.model**

**File name: Book.java**

package com.exercise1.model;  
public class Book {  
 private int id;  
 private String title;  
 private String author;  
 private String isbn;  
 private boolean isAvailable;  
 public Book() {  
 }  
 public Book(int id, String title, String author, String isbn) {  
 this.id = id;  
 this.title = title;  
 this.author = author;  
 this.isbn = isbn;  
 this.isAvailable = true;  
 }  
 public int getId() {  
 return id;  
 }  
 public void setId(int 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 isAvailable;  
 }  
 public void setAvailable(boolean available) {  
 isAvailable = available;  
 @Override  
 public String toString() {  
 return "Book{id=" + id + ", title='" + title + "', author='" + author +  
 "', isbn='" + isbn + "', isAvailable=" + isAvailable + "}";  
 }  
}

**Package name: com.exercise1.service**

**File name: BookService.java**

package com.exercise1.service;

import com.exercise1.model.Book;

import com.exercise1.repository.BookRepository;

import java.util.List

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addNewBook(String title, String author, String isbn) {

Book book = new Book(0, title, author, isbn);

bookRepository.addBook(book);

System.out.println("Service: New book has been added to the library");

public void displayAllBooks() {

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

System.out.println("\n=== Library Book Collection ===");

if (books.isEmpty()) {

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

} else {

for (Book book : books) {

System.out.println(book);

}

}

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

}

public void borrowBook(int bookId) {

Book book = bookRepository.findById(bookId);

if (book != null) {

if (book.isAvailable()) {

book.setAvailable(false);

bookRepository.updateBook(book);

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

} else {

System.out.println("Sorry, this book is already borrowed: " + book.getTitle());

}

} else {

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

}

public void returnBook(int bookId) {

Book book = bookRepository.findById(bookId);

if (book != null) {

if (!book.isAvailable()) {

book.setAvailable(true);

bookRepository.updateBook(book);

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

} else {

System.out.println("This book is already available: " + book.getTitle());

}

} else {

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

}

public void searchBookByTitle(String title) {

Book book = bookRepository.findByTitle(title);

if (book != null) {

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

} else {

System.out.println("No book found with title: " + title);

}

}

public void displayAvailableBooks() {

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

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

if (availableBooks.isEmpty()) {

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

} else {

for (Book book : availableBooks) {

System.out.println(book);

}

}

public void removeBook(int bookId) {

Book book = bookRepository.findById(bookId);

if (book != null) {

bookRepository.deleteBook(bookId);

System.out.println("Service: Book removed from library");

} else {

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

}

}

public void getLibraryStatistics() {

int totalBooks = bookRepository.getTotalBooks();

int availableBooks = bookRepository.findAvailableBooks().size();

int borrowedBooks = totalBooks - availableBooks;

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

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

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

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

}

}

**Package name: com.exercise1.repository**

**File name: BookRepository.java**

package com.exercise1.repository;

import com.exercise1.model.Book;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class BookRepository {

private Map<Integer, Book> books = new HashMap<>();

private int nextId = 1;

public BookRepository() {

addBook(new Book(nextId++, "The Great Gatsby", "F. Scott Fitzgerald", "978-0-7432-7356-5"));

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

addBook(new Book(nextId++, "1984", "George Orwell", "978-0-452-28423-4"));

}

public void addBook(Book book) {

if (book.getId() == 0) {

book.setId(nextId++);

}

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

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

}

public Book findById(int id) {

return books.get(id);

}

public List<Book> findAll() {

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

public Book findByTitle(String title) {

return books.values().stream()

.filter(book -> book.getTitle().equalsIgnoreCase(title))

.findFirst()

.orElse(null);

public void deleteBook(int id) {

Book removedBook = books.remove(id);

if (removedBook != null) {

System.out.println("Book removed: " + removedBook.getTitle());

} else {

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

}

}

public void updateBook(Book book) {

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

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

}

public int getTotalBooks() {

return books.size();

}

public List<Book> findAvailableBooks() {

return books.values().stream()

.filter(Book::isAvailable)

.collect(ArrayList::new, (list, book) -> list.add(book), ArrayList::addAll);

}

}

**Package name:resources**

**File name: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.exercise1.repository.BookRepository"/>

<bean id="bookService" class="com.exercise1.service.BookService">

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

</bean>

</beans>

**File name: 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.exercise1</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

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

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

<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-core</artifactId>

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

</dependency>

<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>11</source>

<target>11</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**File name:LibraryApplication.java**

package com.exercise1;

import com.exercise1.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext

public class LibraryApplication {

public static void main(String[] args) {

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

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

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

System.out.println("\n=== Welcome to Library Management System ===");

bookService.displayAllBooks();

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

bookService.addNewBook("The Catcher in the Rye", "J.D. Salinger", "978-0-316-76948-0");

bookService.searchBookByTitle("1984");

bookService.borrowBook(1);

bookService.displayAvailableBooks()

bookService.returnBook(1);

bookService.getLibraryStatistics();

bookService.addNewBook("Pride and Prejudice", "Jane Austen", "978-0-14-143951-8");

bookService.borrowBook(2);

bookService.borrowBook(4);

bookService.getLibraryStatistics();

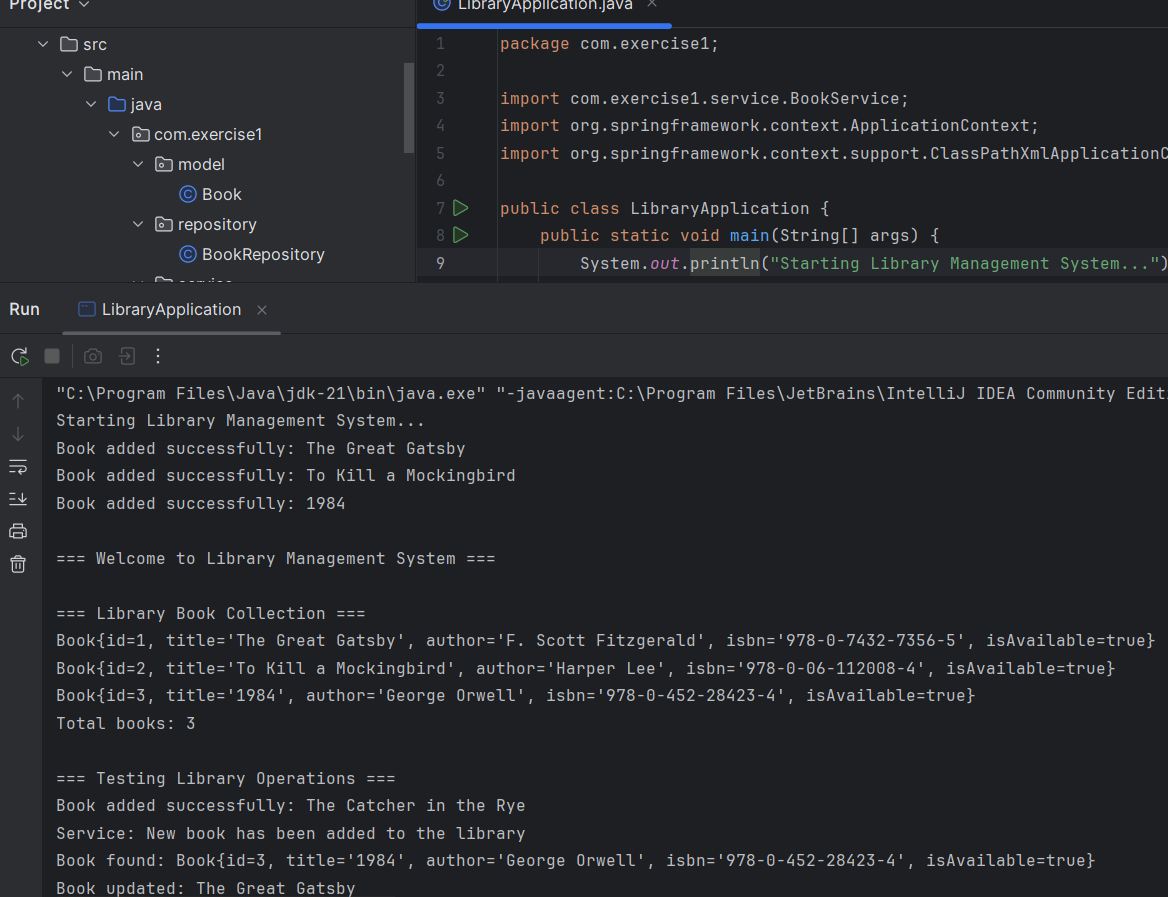
bookService.displayAllBooks();

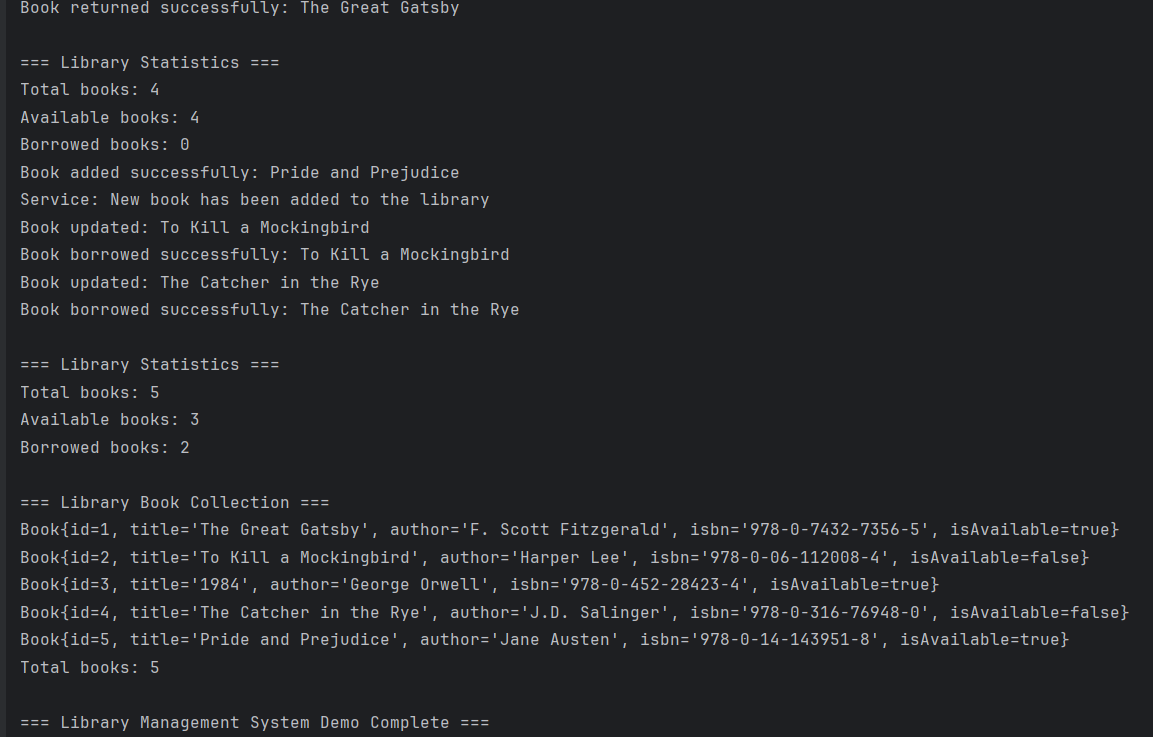
System.out.println("\n=== Library Management System Demo Complete ===");

((ClassPathXmlApplicationContext) context).close();

}}

**Output:**

****

****

**Exercise 2: Implementing Dependency Injection**

**­File name:applicationContext.xml**

**Package name: com.exercise1.model**

**File name: Book.java**

package com.exercise1.model;  
public class Book {  
 private int id;  
 private String title;  
 private String author;  
 private String isbn;  
 private boolean isAvailable;  
 public Book() {  
 }  
 public Book(int id, String title, String author, String isbn) {  
 this.id = id;  
 this.title = title;  
 this.author = author;  
 this.isbn = isbn;  
 this.isAvailable = true;  
 }  
 public int getId() {  
 return id;  
 }  
 public void setId(int 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 isAvailable;  
 }  
 public void setAvailable(boolean available) {  
 isAvailable = available;  
 @Override  
 public String toString() {  
 return "Book{id=" + id + ", title='" + title + "', author='" + author +  
 "', isbn='" + isbn + "', isAvailable=" + isAvailable + "}";  
 }  
}

**Package name: com.exercise1.service**

**File name: BookService.java**

package com.exercise1.service;

import com.exercise1.model.Book;

import com.exercise1.repository.BookRepository;

import java.util.List;

public class BookService {

private BookRepository bookRepository;

public BookService() {

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

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

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

}

public BookRepository getBookRepository() {

return bookRepository;

}

public void addNewBook(String title, String author, String isbn) {

validateDependency();

Book book = new Book(0, title, author, isbn);

bookRepository.addBook(book);

System.out.println("Service: New book has been added to the library");

}

public void displayAllBooks() {

validateDependency();

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

System.out.println("\n=== Library Book Collection ===");

if (books.isEmpty()) {

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

} else {

for (Book book : books) {

System.out.println(book);

}

}

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

}

public void borrowBook(int bookId) {

validateDependency();

Book book = bookRepository.findById(bookId);

if (book != null) {

if (book.isAvailable()) {

book.setAvailable(false);

bookRepository.updateBook(book);

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

} else {

System.out.println("Sorry, this book is already borrowed: " + book.getTitle());

}

} else {

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

}

}

public void returnBook(int bookId) {

validateDependency();

Book book = bookRepository.findById(bookId);

if (book != null) {

if (!book.isAvailable()) {

book.setAvailable(true);

bookRepository.updateBook(book);

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

} else {

System.out.println("This book is already available: " + book.getTitle());

}

} else {

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

}

}

public void searchBookByTitle(String title) {

validateDependency();

Book book = bookRepository.findByTitle(title);

if (book != null) {

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

} else {

System.out.println("No book found with title: " + title);

}

}

public void displayAvailableBooks() {

validateDependency();

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

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

if (availableBooks.isEmpty()) {

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

} else {

for (Book book : availableBooks) {

System.out.println(book);

}

}

}

public void removeBook(int bookId) {

validateDependency();

Book book = bookRepository.findById(bookId);

if (book != null) {

bookRepository.deleteBook(bookId);

System.out.println("Service: Book removed from library");

} else {

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

}

}

public void getLibraryStatistics() {

validateDependency();

int totalBooks = bookRepository.getTotalBooks();

int availableBooks = bookRepository.findAvailableBooks().size();

int borrowedBooks = totalBooks - availableBooks;

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

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

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

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

}

private void validateDependency() {

if (bookRepository == null) {

throw new IllegalStateException("BookRepository dependency not injected!");

}

}

public void testDependencyInjection() {

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

if (bookRepository != null) {

System.out.println("SUCCESS: BookRepository is properly injected");

System.out.println("Repository class: " + bookRepository.getClass().getSimpleName());

} else {

System.out.println("FAILED: BookRepository is not injected");

}

}

}

**Package name: com.exercise1.service**

**File name: LibraryManager.java**

package com.exercise1.service;

import com.exercise1.repository.BookRepository;

public class LibraryManager {

private BookService bookService;

private BookRepository bookRepository;

public LibraryManager(BookService bookService, BookRepository bookRepository) {

this.bookService = bookService;

this.bookRepository = bookRepository;

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

public void performLibraryOperations() {

System.out.println("\n=== LibraryManager Operations ===");

testDependencyInjection();

bookService.addNewBook("Java Programming", "Oracle Press", "978-1-25-963817-4");

bookService.displayAllBooks();

bookService.borrowBook(1);

bookService.getLibraryStatistics();

bookRepository.displayRepositoryInfo();

}

private void testDependencyInjection() {

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

if (bookService != null && bookRepository != null) {

System.out.println("SUCCESS: Both dependencies injected via constructor");

System.out.println("BookService: " + bookService.getClass().getSimpleName());

System.out.println("BookRepository: " + bookRepository.getClass().getSimpleName());

} else {

System.out.println("FAILED: Constructor injection not working");

}

}

public BookService getBookService() {

return bookService;

}

public BookRepository getBookRepository() {

return bookRepository;

}

}

**Package name: com.exercise1.repository**

**File name: BookRepository.java**

package com.exercise1.repository;

import com.exercise1.model.Book;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class BookRepository {

private Map<Integer, Book> books = new HashMap<>();

private int nextId = 1;

public BookRepository() {

System.out.println("BookRepository instance created");

initializeBooks();

}

private void initializeBooks() {

addBook(new Book(nextId++, "The Great Gatsby", "F. Scott Fitzgerald", "978-0-7432-7356-5"));

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

addBook(new Book(nextId++, "1984", "George Orwell", "978-0-452-28423-4"));

System.out.println("Repository initialized with " + books.size() + " books");

}

public void addBook(Book book) {

if (book.getId() == 0) {

book.setId(nextId++);

}

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

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

}

public Book findById(int id) {

return books.get(id);

}

public List<Book> findAll() {

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

}

public Book findByTitle(String title) {

return books.values().stream()

.filter(book -> book.getTitle().equalsIgnoreCase(title))

.findFirst()

.orElse(null);

}

public void deleteBook(int id) {

Book removedBook = books.remove(id);

if (removedBook != null) {

System.out.println("Book removed: " + removedBook.getTitle());

} else {

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

}

}

public void updateBook(Book book) {

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

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

}

public int getTotalBooks() {

return books.size();

}

public List<Book> findAvailableBooks() {

return books.values().stream()

.filter(Book::isAvailable)

.collect(ArrayList::new, (list, book) -> list.add(book), ArrayList::addAll);

}

public void displayRepositoryInfo() {

System.out.println("Repository contains " + books.size() + " books in memory");

}

}

**Package name:resources**

**File name: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.exercise1.repository.BookRepository"/>

<bean id="bookService" class="com.exercise1.service.BookService">

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

</bean>

<bean id="libraryManager" class="com.exercise1.service.LibraryManager">

<constructor-arg ref="bookService"/>

<constructor-arg ref="bookRepository"/>

</bean>

</beans>

**File name: 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.exercise1</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

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

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

<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-core</artifactId>

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

</dependency>

<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>11</source>

<target>11</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**File name:LibraryApplication.java**

package com.exercise1;

import com.exercise1.service.BookService;

import com.exercise1.service.LibraryManager;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryApplication {

public static void main(String[] args) {

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

System.out.println("Initializing Spring IoC Container...");

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

System.out.println("\n=== Spring Container Loaded Successfully ===");

demonstrateSetterInjection(context);

demonstrateConstructorInjection(context);

runLibraryOperations(context);

System.out.println("\n=== Library Management System Demo Complete ===");

((ClassPathXmlApplicationContext) context).close();

}

private static void demonstrateSetterInjection(ApplicationContext context) {

System.out.println("\n=== Demonstrating Setter-based Dependency Injection ===");

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

bookService.testDependencyInjection();

bookService.displayAllBooks();

}

private static void demonstrateConstructorInjection(ApplicationContext context) {

System.out.println("\n=== Demonstrating Constructor-based Dependency Injection ===");

LibraryManager libraryManager = context.getBean("libraryManager", LibraryManager.class);

libraryManager.performLibraryOperations();

}

private static void runLibraryOperations(ApplicationContext context) {

System.out.println("\n=== Running Complete Library Operations ===");

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

bookService.addNewBook("Spring Framework Guide", "Spring Team", "978-1-59-059631-2");

bookService.searchBookByTitle("1984");

bookService.borrowBook(2);

bookService.displayAvailableBooks();

bookService.returnBook(1);

bookService.getLibraryStatistics();

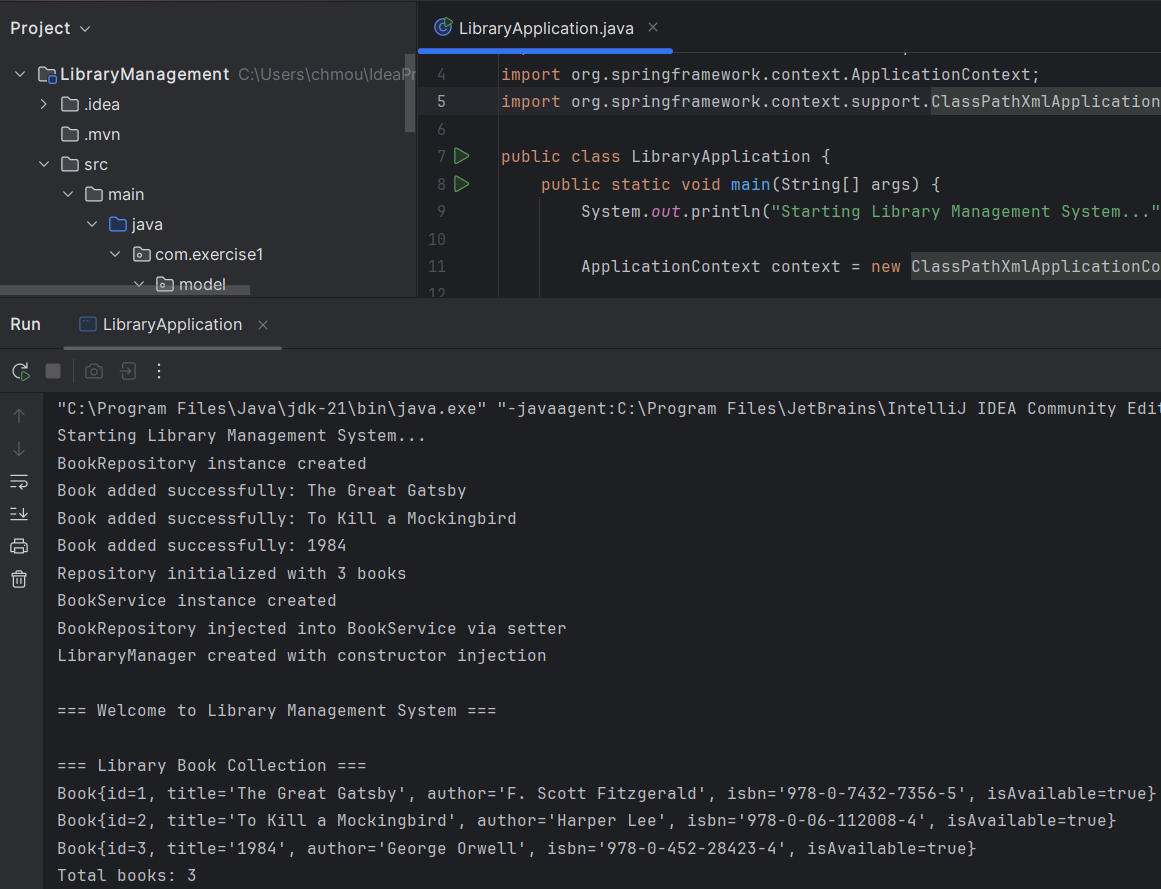
bookService.removeBook(4);

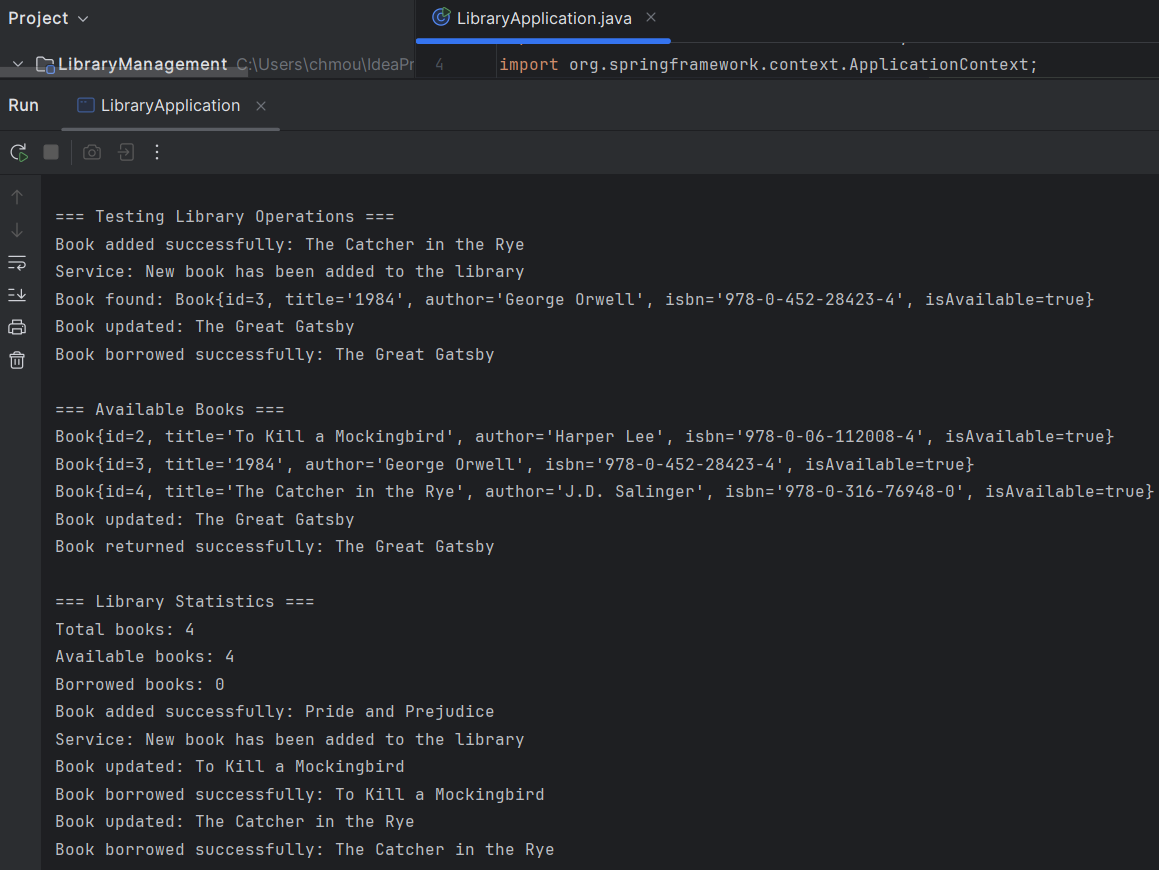
bookService.displayAllBooks();

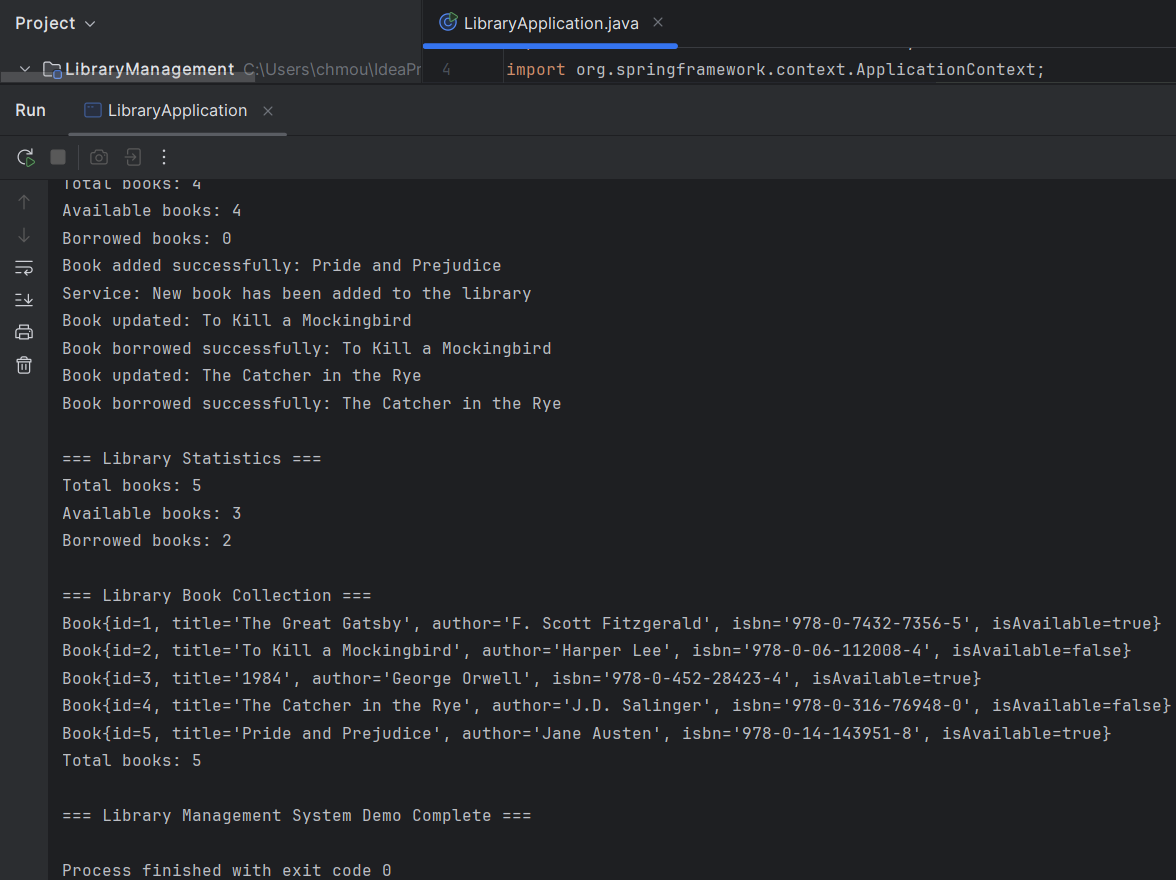
}

}

**Output:**

****

****

****

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

**File name: 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>war</packaging>

<name>Library Management System</name>

<description>A comprehensive library management application built with Spring Framework</description>

<properties>

<maven.compiler.source>21</maven.compiler.source>

<maven.compiler.target>21</maven.compiler.target>

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

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

<junit.version>4.13.2</junit.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>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

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

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

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

<version>4.0.1</version>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

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

<version>2.3.3</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.33</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.36</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.12</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<finalName>library-management</finalName>

<plugins>

<plugin>

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

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

<version>3.8.1</version>

<configuration>

<source>21</source>

<target>21</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

<plugin>

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

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

<version>3.2.3</version>

<configuration>

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

<failOnMissingWebXml>false</failOnMissingWebXml>

</configuration>

</plugin>

<plugin>

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

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

<version>3.0.0-M7</version>

<configuration>

<includes>

<include>\*\*/\*Test.java</include>

<include>\*\*/\*Tests.java</include>

</includes>

</configuration>

</plugin>

<plugin>

<groupId>org.eclipse.jetty</groupId>

<artifactId>jetty-maven-plugin</artifactId>

<version>9.4.48.v20220622</version>

<configuration>

<scanIntervalSeconds>10</scanIntervalSeconds>

<webApp>

<contextPath>/library</contextPath>

</webApp>

<httpConnector>

<port>8080</port>

</httpConnector>

</configuration>

</plugin>

</plugins>

</build>

</project>

**File name: LibraryManagementApplication.java**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

@Configuration

@ComponentScan(basePackages = "com.library")

public class LibraryManagementApplication {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(LibraryManagementApplication.class);

System.out.println("Library Management System is starting up...");

System.out.println("Spring context loaded successfully with " + context.getBeanDefinitionCount() + " beans");

String[] beanNames = context.getBeanDefinitionNames();

System.out.println("Available beans:");

for (String beanName : beanNames) {

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

}

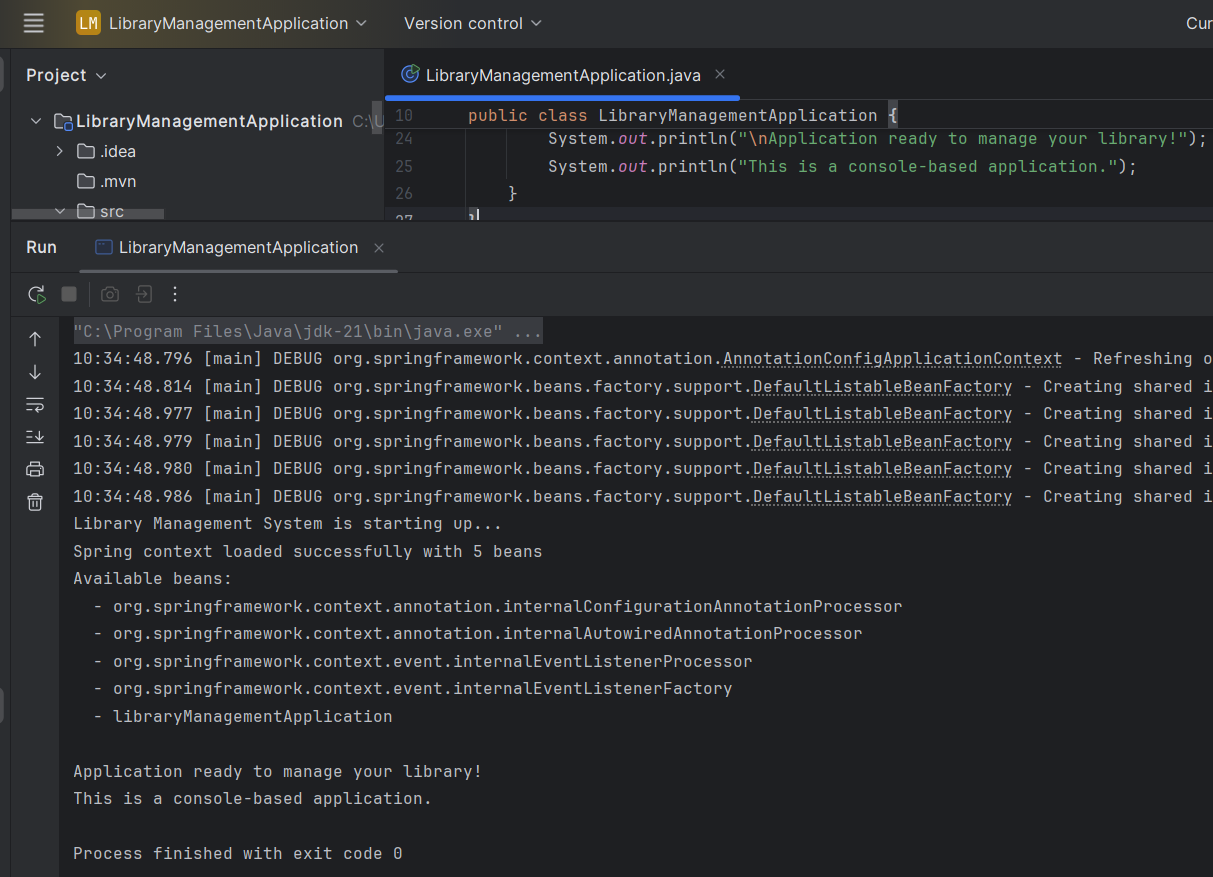
System.out.println("\nApplication ready to manage your library!");

System.out.println("This is a console-based application.");

}

}

**Output:**

****