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

**Book.java**

package com.library.model;

public class Book {

private int id;

private String title;

private String author;

private boolean available;

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

this.id = id;

this.title = title;

this.author = author;

this.available = available;

}

public int getId() { return id; }

public String getTitle() { return title; }

public String getAuthor() { return author; }

public boolean isAvailable() { return available; }

public void setAvailable(boolean available) { this.available = available; }

@Override

public String toString() {

return id + ": " + title + " by " + author + (available ? " (Available)" : " (Not Available)");

}

}

**BookRepository.java**

package com.library.repository;

import com.library.model.Book;

import java.util.List;

public interface BookRepository {

List<Book> findAll();

}

**BookRepositoryImpl.java**

package com.library.repository;

import com.library.model.Book;

import java.util.ArrayList;

import java.util.List;

public class BookRepositoryImpl implements BookRepository {

private final List<Book> books = new ArrayList<>();

public BookRepositoryImpl() {

books.add(new Book(1, "Effective Java", "Joshua Bloch", true));

books.add(new Book(2, "Clean Code", "Robert C. Martin", true));

books.add(new Book(3, "Refactoring", "Martin Fowler", false));

}

@Override

public List<Book> findAll() {

return books;

}

}

**BookService.java**

package com.library.service;

public interface BookService {

void displayBooks();

}

**BookServiceImpl.java**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

import org.junit.Before;

import org.junit.Test;

import java.util.Arrays;

import java.util.List;

import static org.mockito.Mockito.\*;

public class BookServiceImplTest {

private BookRepository mockRepository;

private BookServiceImpl bookService;

@Before

public void setUp() {

mockRepository = mock(BookRepository.class);

bookService = new BookServiceImpl();

bookService.setBookRepository(mockRepository);

}

@Test

public void testDisplayBooks() {

List<Book> mockBooks = Arrays.asList(

new Book(1, "Effective Java", "Joshua Bloch", true),

new Book(2, "Clean Code", "Robert C. Martin", true)

);

when(mockRepository.findAll()).thenReturn(mockBooks);

bookService.displayBooks();

verify(mockRepository, times(1)).findAll();

}

}

**BookServiceImplTest.java**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

import org.junit.Before;

import org.junit.Test;

import java.util.Arrays;

import java.util.List;

import static org.mockito.Mockito.\*;

public class BookServiceImplTest {

private BookRepository mockRepository;

private BookServiceImpl bookService;

@Before

public void setUp() {

mockRepository = mock(BookRepository.class);

bookService = new BookServiceImpl();

bookService.setBookRepository(mockRepository);

}

@Test

public void testDisplayBooks() {

List<Book> mockBooks = Arrays.asList(

new Book(1, "Effective Java", "Joshua Bloch", true),

new Book(2, "Clean Code", "Robert C. Martin", true)

);

when(mockRepository.findAll()).thenReturn(mockBooks);

bookService.displayBooks();

verify(mockRepository, times(1)).findAll();

}

}

1. **[gnizant-Digital-Nurture-4.0-JavaFSE-Handson-2025](https://github.com/Thanuja2102/Cognizant-Digital-Nurture-4.0-JavaFSE-Handson-2025/tree/main)**
2. /[Week-3 SpringCore\_Maven](https://github.com/Thanuja2102/Cognizant-Digital-Nurture-4.0-JavaFSE-Handson-2025/tree/main/Week-3 SpringCore_Maven)
3. /[EX5 Configuring the Spring IoC Container](https://github.com/Thanuja2102/Cognizant-Digital-Nurture-4.0-JavaFSE-Handson-2025/tree/main/Week-3 SpringCore_Maven/EX5 Configuring the Spring IoC Container)

/

# **BorrowingServiceImpl.java**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

import java.util.List;

public class BorrowingServiceImpl {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void borrowBookById(int bookId) {

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

for (Book book : books) {

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

if (book.isAvailable()) {

book.setAvailable(false);

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

} else {

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

}

return;

}

}

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

}

}

**BorrowingServiceImplTest.java**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

import org.junit.Before;

import org.junit.Test;

import java.util.Arrays;

import static org.mockito.Mockito.\*;

import static org.junit.Assert.\*;

public class BorrowingServiceImplTest {

private BookRepository mockRepository;

private BorrowingServiceImpl borrowingService;

@Before

public void setUp() {

mockRepository = mock(BookRepository.class);

borrowingService = new BorrowingServiceImpl();

borrowingService.setBookRepository(mockRepository);

}

@Test

public void testBorrowAvailableBook() {

Book book = new Book(1, "Clean Architecture", "Robert C. Martin", true);

when(mockRepository.findAll()).thenReturn(Arrays.asList(book));

borrowingService.borrowBookById(1);

assertFalse(book.isAvailable());

}

@Test

public void testBorrowAlreadyBorrowedBook() {

Book book = new Book(2, "Clean Code", "Robert C. Martin", false);

when(mockRepository.findAll()).thenReturn(Arrays.asList(book));

borrowingService.borrowBookById(2);

assertFalse(book.isAvailable());

}

@Test

public void testBorrowNonExistentBook() {

when(mockRepository.findAll()).thenReturn(Arrays.asList());

borrowingService.borrowBookById(100);

}

}

**LibraryApp.java**

package com.library.main;

import com.library.repository.BookRepositoryImpl;

import com.library.service.BookServiceImpl;

import com.library.service.BorrowingServiceImpl;

public class LibraryApp {

public static void main(String[] args) {

BookRepositoryImpl repo = new BookRepositoryImpl();

BookServiceImpl bookService = new BookServiceImpl();

BorrowingServiceImpl borrowService = new BorrowingServiceImpl();

bookService.setBookRepository(repo);

borrowService.setBookRepository(repo);

System.out.println("All Books:");

bookService.displayBooks();

borrowService.borrowBookById(2);

borrowService.borrowBookById(2);

borrowService.borrowBookById(99);

System.out.println("After Borrowing:");

bookService.displayBooks();

}

}

**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>Spring\_IOC</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>22</maven.compiler.source>

<maven.compiler.target>22</maven.compiler.target>

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

</properties>

<dependencies>

<!-- Spring Core (for IoC) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring AOP (optional if using AOP) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<!-- AspectJ (required for Spring AOP) -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.21.1</version>

</dependency>

<!-- JUnit for Unit Testing -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

<!-- Mockito for mocking in unit tests -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.11.0</version>

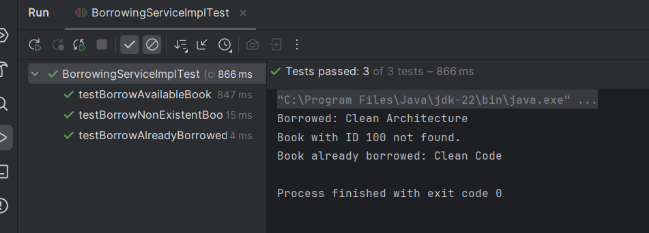
<scope>test</scope>

</dependency>

</dependencies>

</project>

**OUTPUT :**

****