**Digital Nurture 4.0 – Week 3**

**(i) Spring Core and Maven**

**(a) Mandatory hands-on**

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

**Step 1:**

Launch Eclipse IDE with new Maven Project named **LibraryManagement**.

**Filled details:**

Group Id: com.library

Artifact Id: LibraryManagement

Packaging: jar

**Step 2:**

Add dependencies in pom.xml.

**Pom.xml:**

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

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

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

https://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>

<properties>

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

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

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Step 3:**

In **src/main/java**,create a package named **com.library.repository,com.library** and **com.library.service.**Then create a classes named BookRepository.java on Repository folder,LibraryManagementApplication.java on library folder and BookService.java on service folder.

**BookRepository.java:**

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.out.println("BookRepository: Name of the Saving book - " + bookName);

}

}

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.out.println("BookService: Name of the Adding book - " + bookName);

bookRepository.saveBook(bookName);

}

}

**LibraryManagementApplication.java:**

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) {

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

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

bookService.addBook("Harry Potter");

}

}

**Step 4:**

In **src/main/resources**,create **applicationContext.xml.**

**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

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

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

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

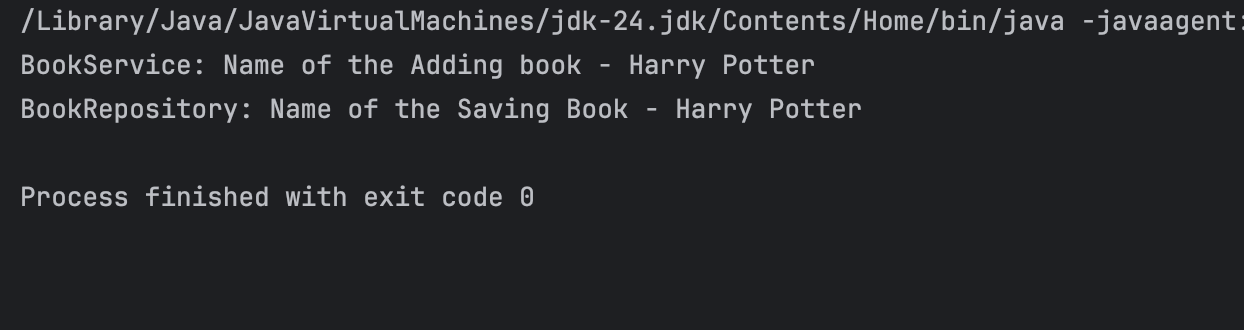
<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

**Step 5:**

Run LibraryManagementApplication.java as java application.



**Exercise 2: Implementing Dependency Injection:**

**Step 1:**

Modify the Spring XML to actually wire BookRepository into BookService.

**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

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

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

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

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

</bean>

</beans>

**Step 2:**

Ensure BookService Has Setter Method.

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.out.println("BookService: Name of the Adding book - " + bookName);

bookRepository.saveBook(bookName);

}

}

**Step 3:**

Test the Configuration in Eclipse**.**

**LibraryManagementApplication.java:**

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) {

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

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

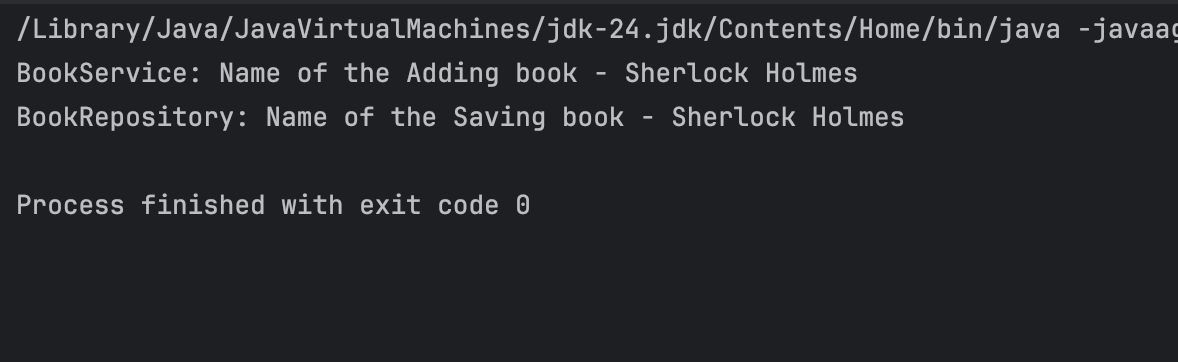
bookService.addBook("Sherlock Holmes");

}

}

**Step 4:**

Again Run LibraryManagementApplication.java as java application.



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

**Step 1:**

Create a new Maven project named LibraryManagement.

**Filled details:**

Group Id: com.library

Artifact Id: LibraryManagement

Packaging: jar

**Step 2:**

In pom.xml,

**Pom.xml:**

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

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

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

https://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>

<properties>

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

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

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>5.3.30</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.30</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

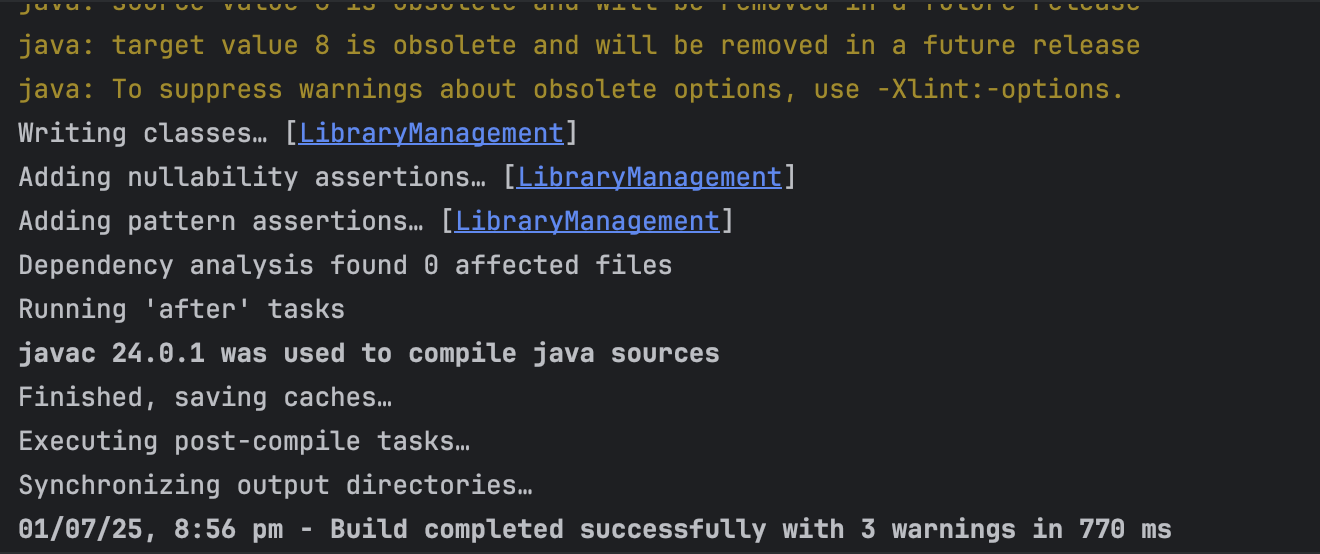
</plugins>

</build>

</project>

**Step 3:**

Force Maven to Download Dependencies.Finally it will shows Build completing status.



**(b) Additional important hands-on**

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

**Step 1:**

Update applicationContext.xml in the **src/main/resources** directory.

**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

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

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

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

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

</bean>

</beans>

**Step 2:**

Update BookService.java in the **src/main/java/com/library/service** directory.

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.out.println("BookService: Name of the Adding book - " + bookName);

bookRepository.saveBook(bookName);

}

}

**Step 3:**

Update BookRepository.java in the **src/main/java/com/library/repository** directory.

**BookRepository.java:**

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.out.println("BookRepository: Name of the Saving book - " + bookName);

}

}

**Step 4:**

Create Main Class named LibraryManagementApplication.java in the **src/main/java/com/library** directory.

**LibraryManagementApplication.java:**

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) {

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

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

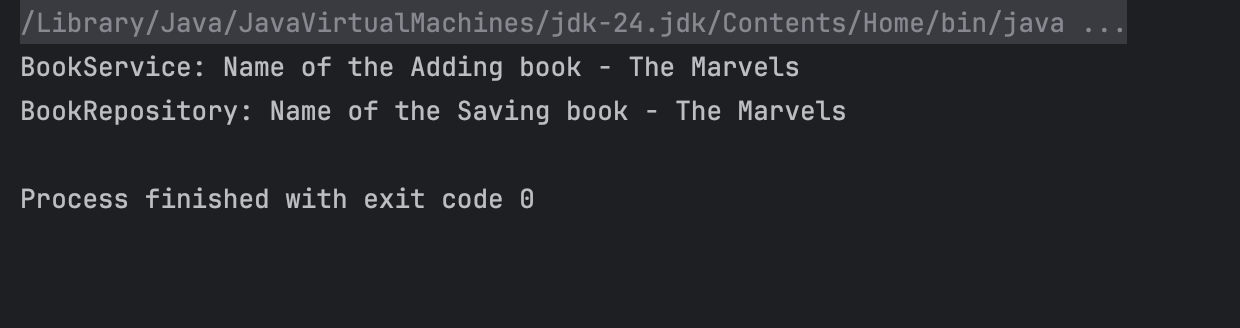
bookService.addBook("The Marvels");

}

}

**Step 5:**

Run LibraryManagementApplication.java as java application.



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

**Step 1:**

Update applicationContext.xml in the **src/main/resources** directory.

**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

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

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

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

<constructor-arg ref="bookRepository"/>

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

</bean>

</beans>

**Step 2:**

Update BookService.java in the **src/main/java/com/library/service** directory.

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public BookService(BookRepository bookRepository) {

System.out.println("Constructor Injection: BookRepository was injected");

this.bookRepository = bookRepository;

}

public void setBookRepository(BookRepository bookRepository) {

System.out.println("Setter Injection: BookRepository was injected");

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.out.println("BookService: Name of the Adding book - " + bookName);

bookRepository.saveBook(bookName);

}

}

**Step 3:**

Update BookRepository.java in the **src/main/java/com/library/repository** directory.(No chnages needed in **BookRepository.java**)

**BookRepository.java:**

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.out.println("BookRepository: Name of the Saving book - " + bookName);

}

}

**Step 4:**

Create Main Class named LibraryManagementApplication.java in the **src/main/java/com/library** directory.

**LibraryManagementApplication.java:**

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) {

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

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

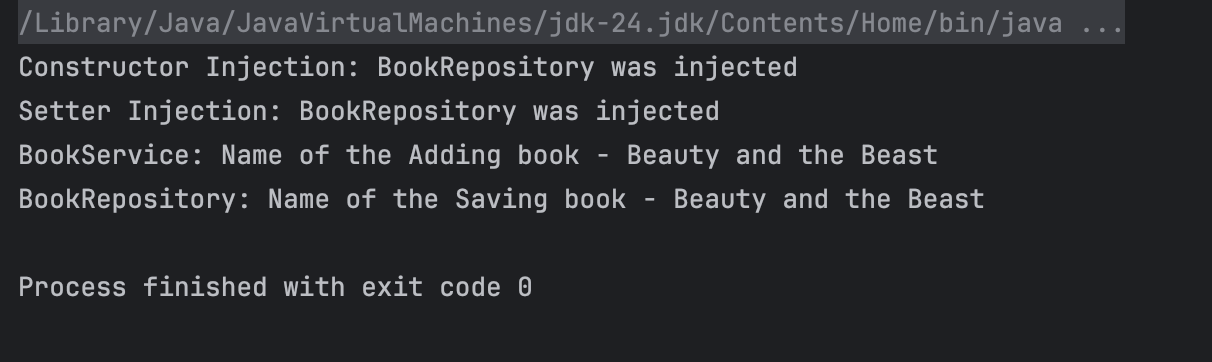
bookService.addBook("Beauty and the Beast");

}

}

**Step 5:**

Run LibraryManagementApplication.java as java application.



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

**Step 1:**

Go to Spring Initializr(https://start.spring.io) and select (or) enter

Project: Maven  
Language: Java  
Packaging: Jar  
Java Version: 8 or 17 (both will work)  
Name: LibraryManagement  
Group: com.library  
Dependencies:

* Spring Web
* Spring Data JPA
* H2 Database

Finally click the generate button to download a zip file.Then unzip that file and import that into the eclipse idle.

**Step 2:**

In **src/main/resources**,modify the **application.properties** as required

**application.properties**:

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=root

spring.datasource.password=root@123

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

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

From above configuration,change your username and password on your MySQL workbench.

**Step 3:**

Create class named **Book.java** in the **src/main/java/com/library/model** directory.

**Book.java**:

package com.library.model;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

public Book() {

}

public Book(String title, String author) {

this.title = title;

this.author = author;

}

public Long getId() {

return id;

}

public 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;

}

}

**Step 4:**

Create an Interface named **BookRepository.java** in the **src/main/java/com/library/repository** ditectory.

**BookRepository.java**:

package com.library.repository;

import com.library.model.Book;

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

import org.springframework.stereotype.Repository;

@Repository

public interface BookRepository extends JpaRepository<Book, Long> {

}

**Step 5:**

Create a class named **BookController.java** in the **src/main/java/com/library/controller** directory.

**BookController.java**:

package com.library.controller;

import com.library.model.Book;

import com.library.repository.BookRepository;

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

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

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/books")

public class BookController {

@Autowired

private BookRepository bookRepository;

@GetMapping

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

@PostMapping

public Book createBook(@RequestBody Book book) {

return bookRepository.save(book);

}

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

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

return book.orElse(null);

}

@PutMapping("/{id}")

public Book updateBook(@PathVariable Long id, @RequestBody Book bookDetails) {

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

if (optionalBook.isPresent()) {

Book book = optionalBook.get();

book.setTitle(bookDetails.getTitle());

book.setAuthor(bookDetails.getAuthor());

return bookRepository.save(book);

} else {

return null;

}

}

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable Long id) {

bookRepository.deleteById(id);

}

}

**Step 6:**

Update the main class named **LibraryManagementApplication.java** in the **src/main/java/com/library** directory.

**LibraryManagementApplication.java**:

package com.library;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibraryManagementApplication {

public static void main(String[] args) {

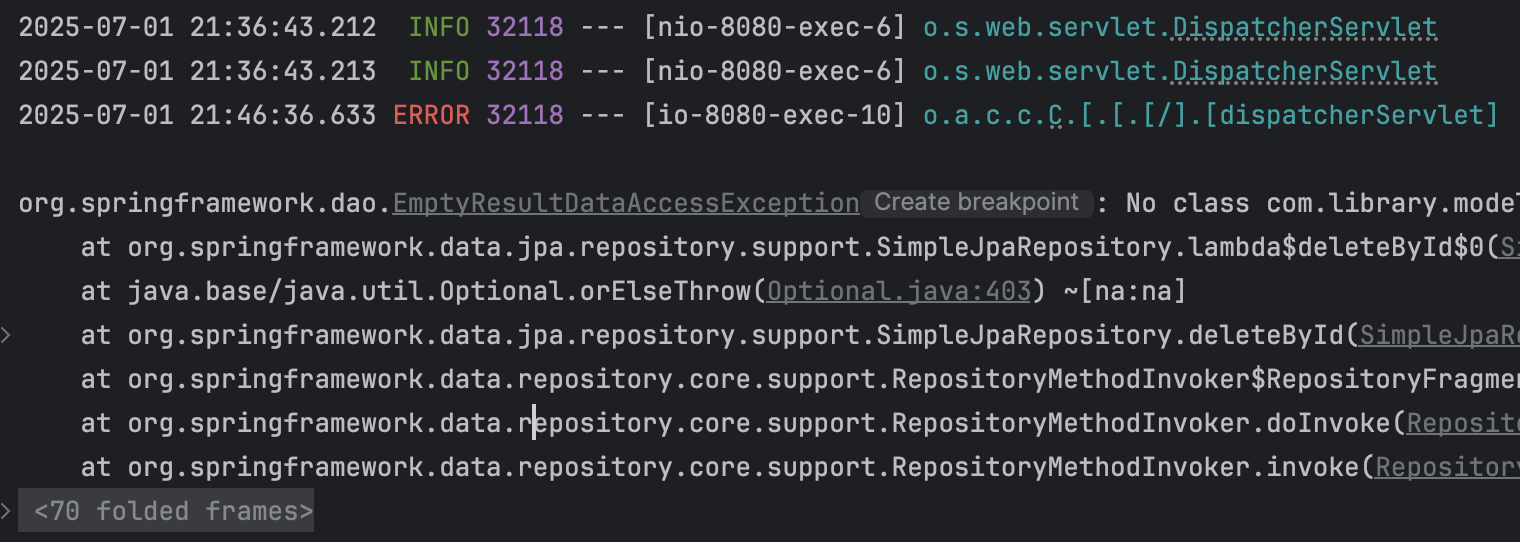
SpringApplication.run(LibraryManagementApplication.class, args);

}

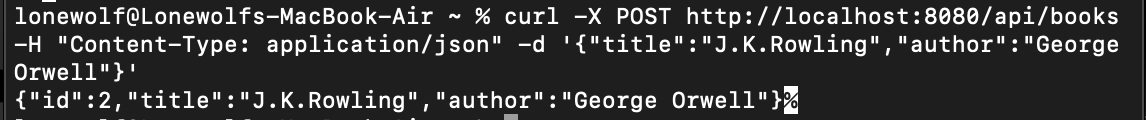
}

**Step 7:**

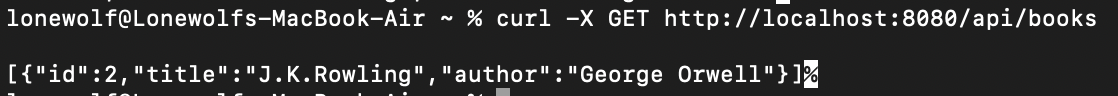
Run LibraryManagementApplication.java as java application.



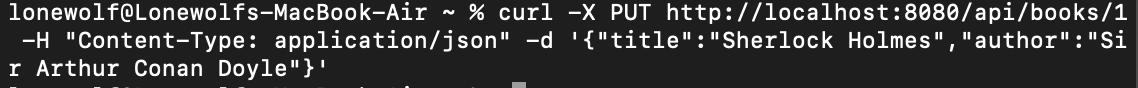
**To Post - new book :**



**To GET- all books:**



**To PUT - update book:**



**To DELETE book:**

