**Digital Nurture 4.0**

Week 3- Spring Core and Maven

# Mandatory HandsOn Exercises

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

**Program:**

* Open Intelli j and create a new maven project,choose the Create from archetype then Enter GroupId: com.library ,ArtifactId: LibraryManagement then create.
* Edit the 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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>untitledLibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>untitledLibraryManagement</name>

<properties>

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

</properties>

<dependencies>

<!-- Spring Core Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</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>

* IntelliJ will ask **"Load Maven Changes?"** - Click **Yes.**
* Then create new packages named com.library.service, com.library.repository, and com.library in the src/main/java.
* Then create class BookRepository.java inside com.library.repository.
* Then create class BookService.java inside com.library.service.
* Then create file LibraryManagementApplication.java inside com.library.

**BookRepository.java:**

package com.library.repository;

public class BookRepository {

public void displayRepo() {

System.out.println("Inside BookRepository");

}

}

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void displayService() {

System.out.println("Inside BookService");

bookRepository.displayRepo();

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

}

**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.displayService();

}

}

* Create applicationContext.xml in the src/main/resources.

**applicationContext.xml:**

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

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

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

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

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

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

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

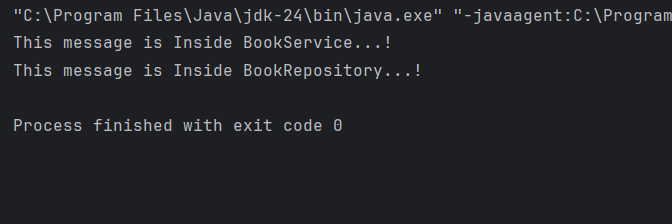
<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

* For running the application,Go to the LibraryManagementApplication.java
* Right-click → Run 'LibraryManagementApplication.main()'

**Output:**



**Exercise 2: Implementing Dependency Injection**

**Program:**

* We already having pom.xml,don’t need to make changes in the pom.xml.
* Modify applicationContext.xml.

**applicationContext.xml:**

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

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

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

</bean>

* Modify the BookService.java.

**BookService.java:**

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public void displayService() {  
 System.*out*.println("Implementing the Dependency Injection in BookService...!");  
 bookRepository.displayRepo();  
 }  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
}

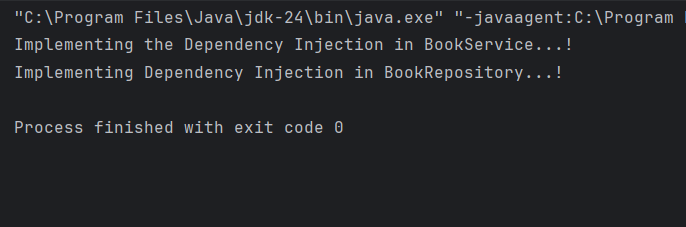
* Modify the BookRepository.java.

**BookRepository.java:**

package com.library.repository;  
  
public class BookRepository {  
 public void displayRepo() {  
 System.*out*.println("Implementing Dependency Injection in BookRepository...!");  
 }  
}

* Don’t need to make changes in the libraryManagementApplication.java,applicationContext.xml.
* For running the application,Go to the LibraryManagementApplication.java
* Right-click → Run 'LibraryManagementApplication.main()'

**Output:**

****

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

**Program:**

* Add Spring & JUnit Dependencies the pom.xml code.

**Pom.xml:**

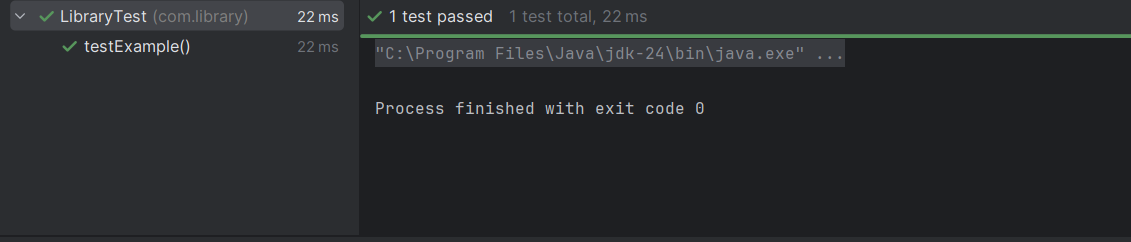
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.library</groupId>  
 <artifactId>untitledLibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>untitledLibraryManagement</name>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.32</version>  
 </dependency>  
  
   
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.32</version>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.32</version>  
 </dependency>  
  
   
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter</artifactId>  
 <version>5.10.2</version>  
 <scope>test</scope>  
 </dependency>  
  
  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>3.8.1</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

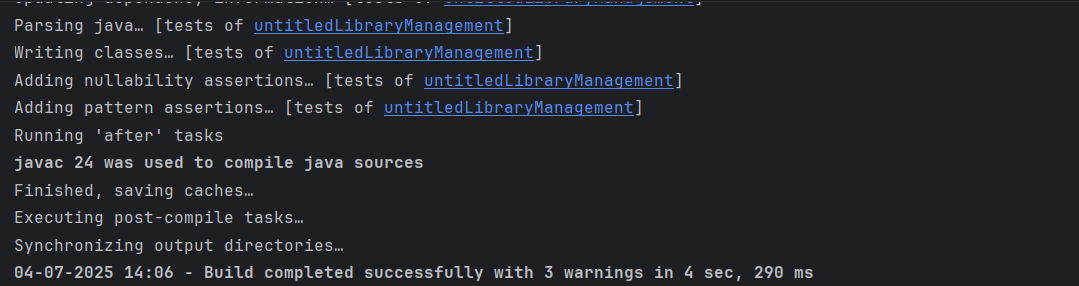
* Reload Maven in IntelliJ.
* Create a test class named LibraryTest.java in src/test/java/com/library/.
* The test class is not necessary you can Build the project directly.

**LibraryTest.java:**

package com.library;  
  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.*assertTrue*;  
  
public class LibraryTest {  
  
 @Test  
 public void testExample() {  
 *assertTrue*(true); // Always passes  
 }  
}

**Output:**

****

****

# Additional important hands-on

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

**Program:**

* Create a new file named applicationContext.xml in src/main/resources.

**applicationContext.xml:**

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

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

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

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

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

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

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

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

</bean>

</beans>

* Add Setter Method in BookService.java.

**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 displayService() {  
 System.*out*.println("BookService is working Perfectly. BookRepository is injected: " + (bookRepository != null));  
 }  
}

**BookRepository.java:**

package com.library.repository;

public class BookRepository {

public void displayRepository() {

System.out.println("BookRepository is working");

}

}

* Create a Main Class to Load Context LibraryManagementApplication.java.

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

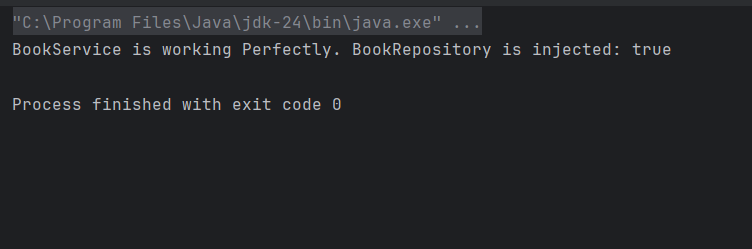
service.displayService();

}

}

* Run the LibraryManagementApplication.java .

**Output:**

****

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

**Program:**

* Just modify the BookService.java.

**BookService.java:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Constructor for constructor injection

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

System.out.println("BookService: Constructor Injection used");

}

// Setter for setter injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

System.out.println("BookService: Setter Injection used");

}

public void displayService() {

System.out.println("BookService is working. BookRepository injected: " + (bookRepository != null));

}

}

* Update applicationContext.xml for both injections in the src/main/resources/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">  
  
 <!-- BookRepository Bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <!-- BookService Bean with constructor and setter injection -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <!-- Constructor Injection -->  
 <constructor-arg ref="bookRepository" />  
 <!-- Setter Injection -->  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

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

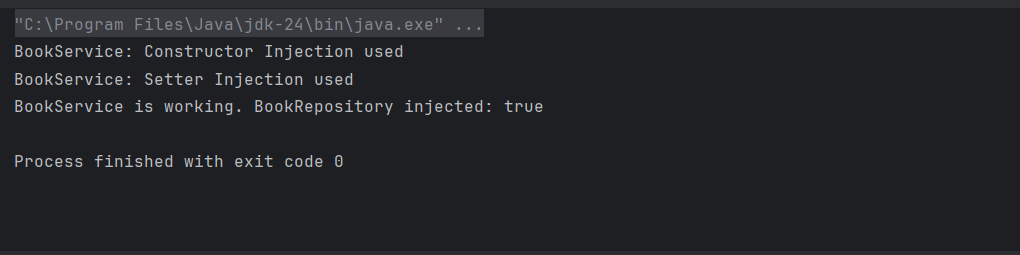
service.displayService();

}

}

* Build the project-Rebuild project.
* Run the LibraryManagementApplication.java.

**Output:**

****

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

**Program:**

**Setting up the intel ij:**

* **Project Type**: Maven
* **Language**: Java
* **Spring Boot Version**: 2.7.18
* **Dependencies**:

Spring Web,Spring Data JPA,MySQL Driver

**MySQL setup:**

* In MySQL workbench Create Schema named librarydb.
* Create the required privileges in src/main/resources/application.properties.

**application.properties:**

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

spring.datasource.username=root

spring.datasource.password=your\_password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

* Modify the 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 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>2.7.18</version>  
 <relativePath/>  
 </parent>  
  
 <groupId>com.library</groupId>  
 <artifactId>librarymanagement</artifactId>  
 <version>1.0</version>  
 <packaging>jar</packaging>  
  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <version>8.0.33</version>  
 </dependency>  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter</artifactId>  
 <version>5.10.2</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>jakarta.persistence</groupId>  
 <artifactId>jakarta.persistence-api</artifactId>  
 <version>2.2.3</version>  
 </dependency>  
 <dependency>  
 <groupId>javax.persistence</groupId>  
 <artifactId>javax.persistence-api</artifactId>  
 <version>2.2</version>  
 </dependency>  
  
 </dependencies>  
  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

* Create Entity in src/main/java/com/library/entity/Book.java.

**Book.java:**

package com.library.entity;  
  
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;  
 private double price;  
  
 // Constructors  
 public Book() {}  
  
 public Book(String title, String author) {  
 this.title = title;  
 this.author = author;  
 }  
  
 // Getters & Setters  
 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;  
 }  
}

* Create Repository in src/main/java/com/library/repository/BookRepository.java

**BookRepository.java:**

package com.library.repository;

import com.library.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Integer> {

}

* Create Controller in src/main/java/com/library/controller/BookController.java

**BookController.java:**

package com.library.controller;

import com.library.entity.Book;

import com.library.repository.BookRepository;

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

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

public class BookController {

@Autowired

private BookRepository repo;

@GetMapping("/books")

public List<Book> getBooks() {

return repo.findAll();

}

@PostMapping("/books")

public Book addBook(@RequestBody Book book) {

return repo.save(book);

}

@PutMapping("/books/{id}")

public Book updateBook(@PathVariable Integer id, @RequestBody Book book) {

book.setId(id);

return repo.save(book);

}

@DeleteMapping("/books/{id}")

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

if (!repo.existsById(id)) {

return ResponseEntity.notFound().build();

}

repo.deleteById(id);

return ResponseEntity.noContent().build();

}

}

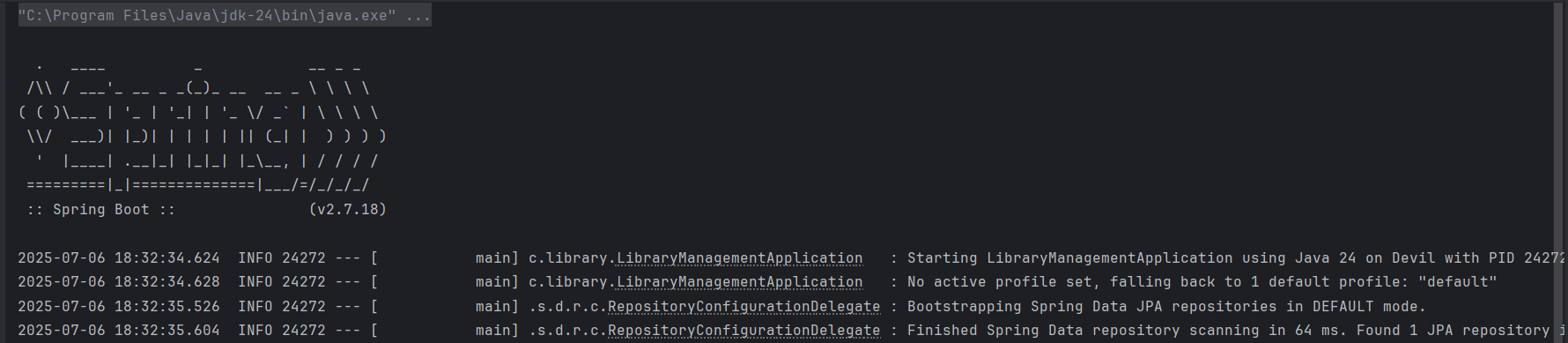
* Create the main class in src/main/java/com/library/LibraryManagementApplication.java

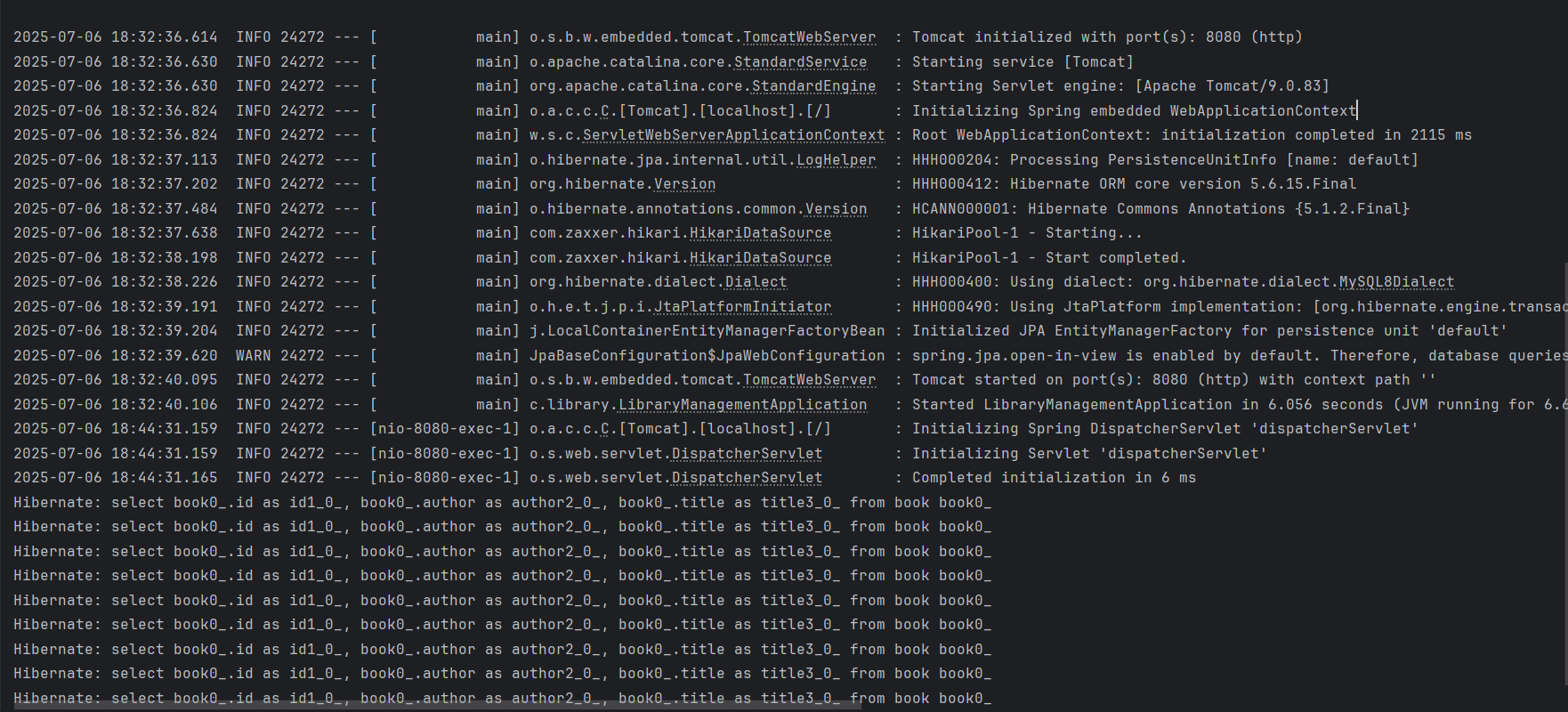
**LibraryManagementApplication.java**

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);  
 }  
}

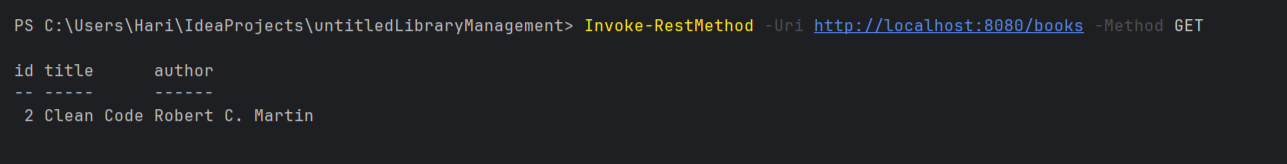
* Run LibraryManagementApplication.java

**Output:**

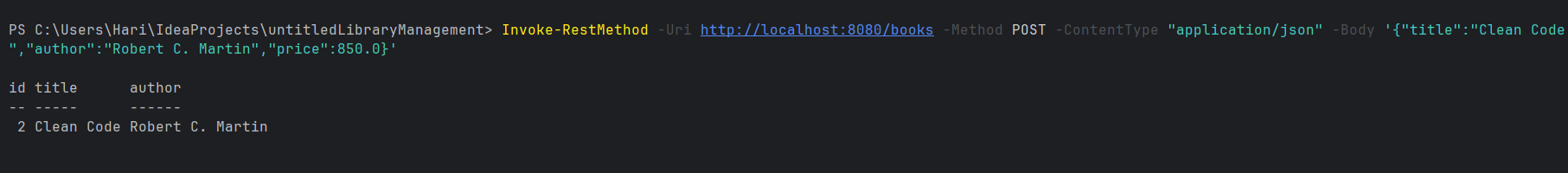
****

****

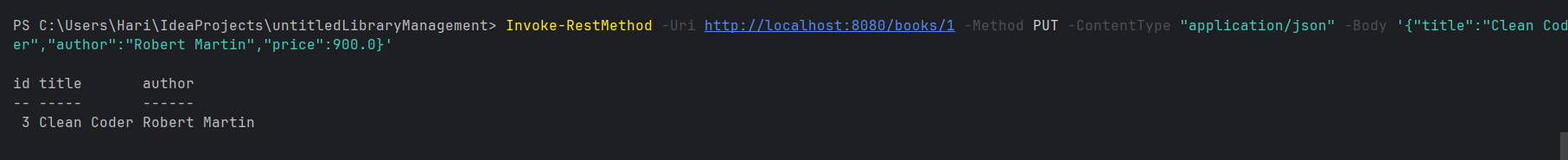
**Create a data:**

****

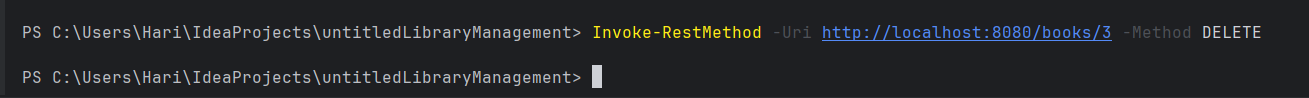
**Read all data:**

****

**Update the data:**

****

**Delete the data:**

****