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

Create a Maven project named **LibraryManagement**

**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>

**BookService**.

package com.library.service;

import java.util.List;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    public *void* setBookRepository(BookRepository *repo*) {

        this.bookRepository = *repo*;

    }

    public List<String> listTitles() {

        return bookRepository.findAllTitles();

    }

}

**BookRepository**

package com.library.repository;

import java.util.Arrays;

import java.util.List;

public class BookRepository {

    public List<String> findAllTitles() {

        return Arrays.asList(

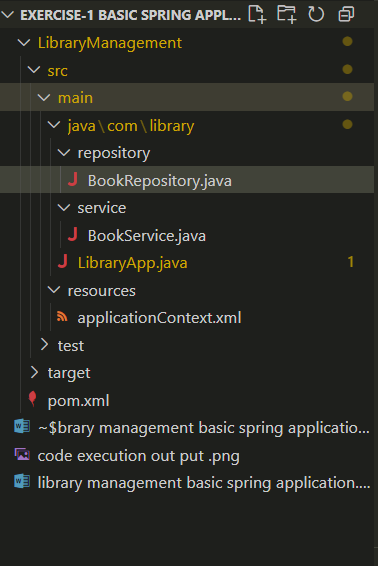
                "Clean Code",

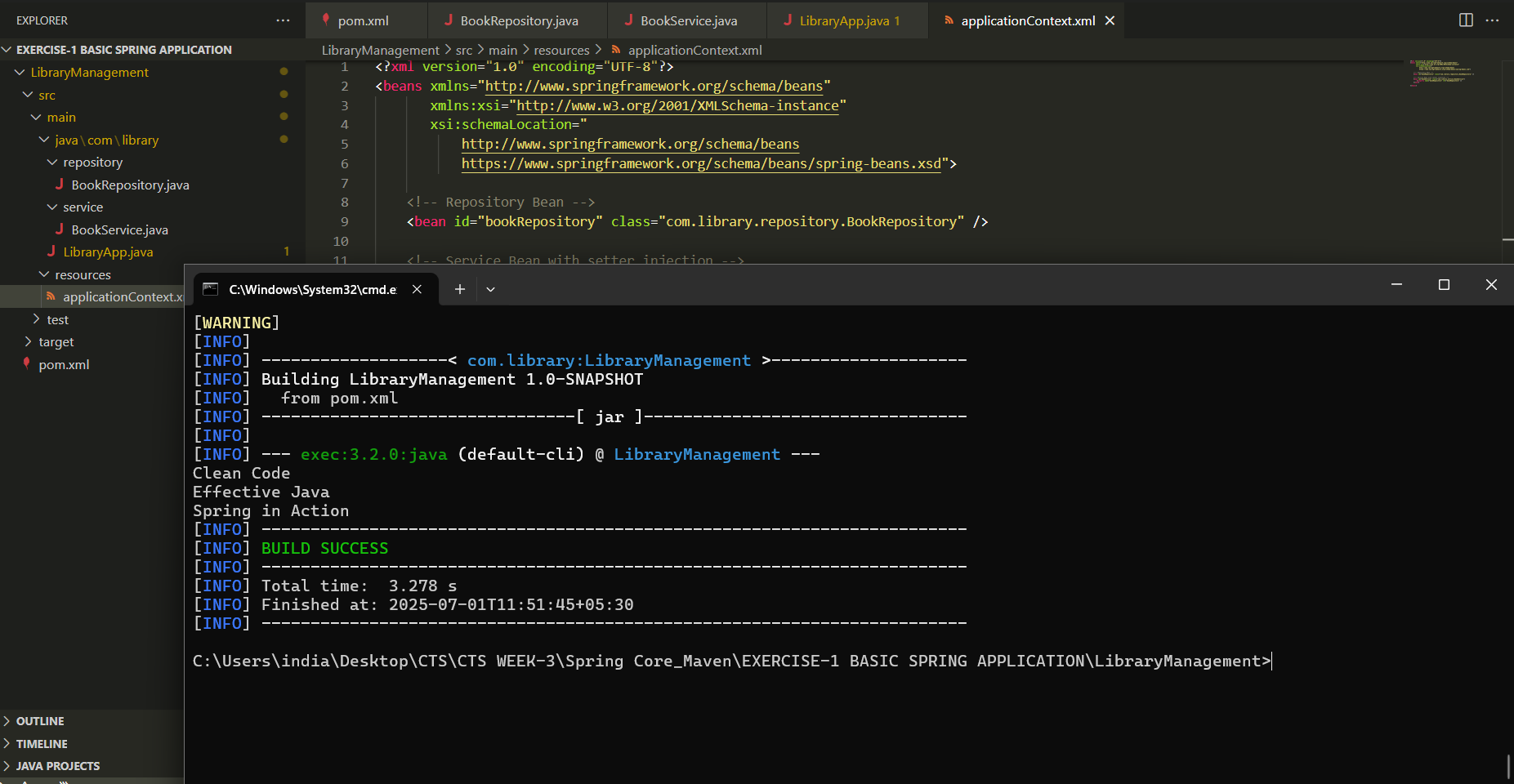
                "Effective Java",

                "Spring in Action");

    }

}

OUTPUT  




**Exercise 2: Implementing Dependency Injection**

**Modify the XML Configuration**

<?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.example.library.repository.BookRepository" />

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

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

    </bean>

</beans>

**BookService Class**

public class BookService {

    private BookRepository bookRepository;

    public *void* setBookRepository(BookRepository *bookRepository*) {

        this.bookRepository = *bookRepository*;

    }

    public *void* addBook(Book *book*) {

        bookRepository.save(*book*);

    }

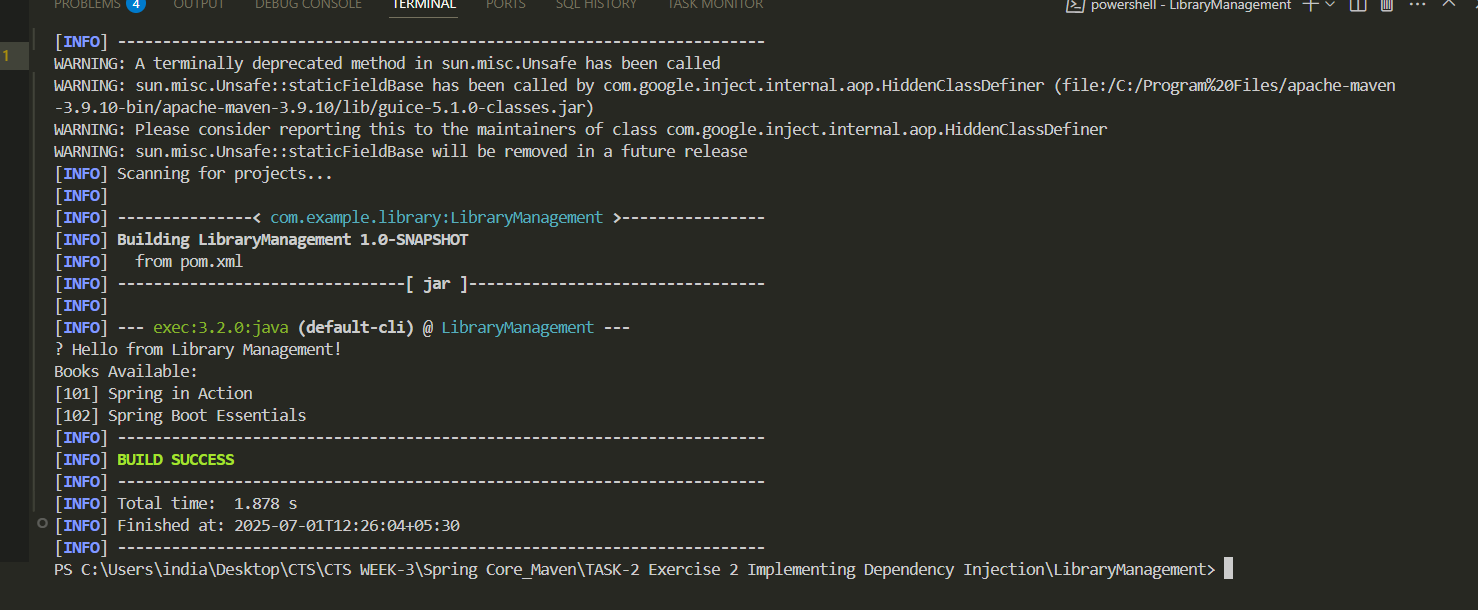
    public List<Book> getAllBooks() {

        return bookRepository.findAll();

    }

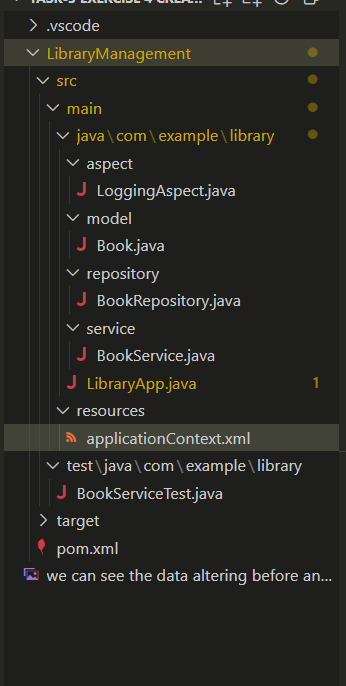
}

**Test the Configuration**



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

**Create a New Maven Project:**



**Add Spring Dependencies in pom.xml:**

 <dependency>

      <groupId>org.springframework</groupId>

      <artifactId>spring-context</artifactId>

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

    </dependency>

<dependency>

      <groupId>org.springframework</groupId>

      <artifactId>spring-webmvc</artifactId>

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

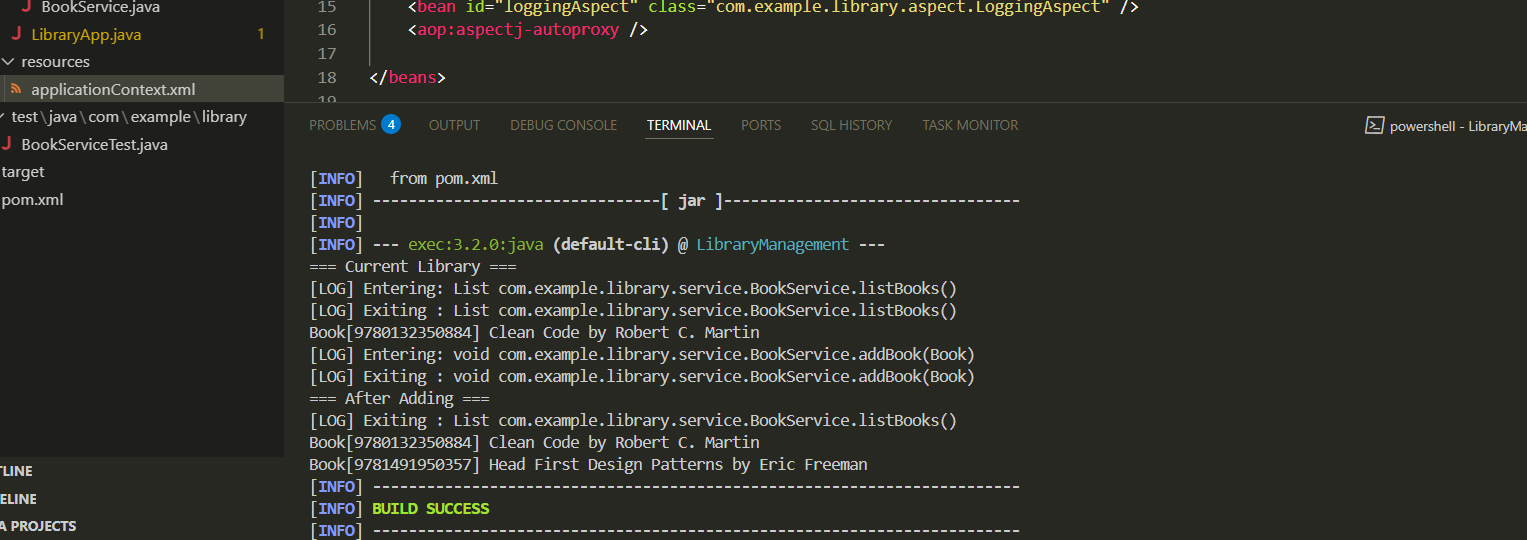
    </dependency>

**Configure Maven Plugins:**

* + Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

1. <properties>
2. <java.version>1.8</java.version>
3. <spring.version>5.3.39</spring.version> <!-- last Spring 5 LTS that supports Java 8 -->
4. <maven.compiler.plugin.version>3.12.0</maven.compiler.plugin.version>
5. <maven.exec.plugin.version>3.2.0</maven.exec.plugin.version>
6. <maven.surefire.plugin.version>3.2.5</maven.surefire.plugin.version>
7. <junit.jupiter.version>5.10.2</junit.jupiter.version>
8. </properties>

**output**



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

Update applicationContext.**xml** to configure constructor injection for **BookService**.

<?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.example.library.repository.BookRepository"/>

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

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

    </bean>

</beans>

Ensure that the **BookService** class has a setter method for **BookRepository** and configure it in **applicationContext.xml**.

package com.example.library.service;

import com.example.library.repository.BookRepository;

import java.util.List;

public class BookService {

    private BookRepository bookRepository;

    public *void* setBookRepository(BookRepository *bookRepository*) {

        this.bookRepository = *bookRepository*;

    }

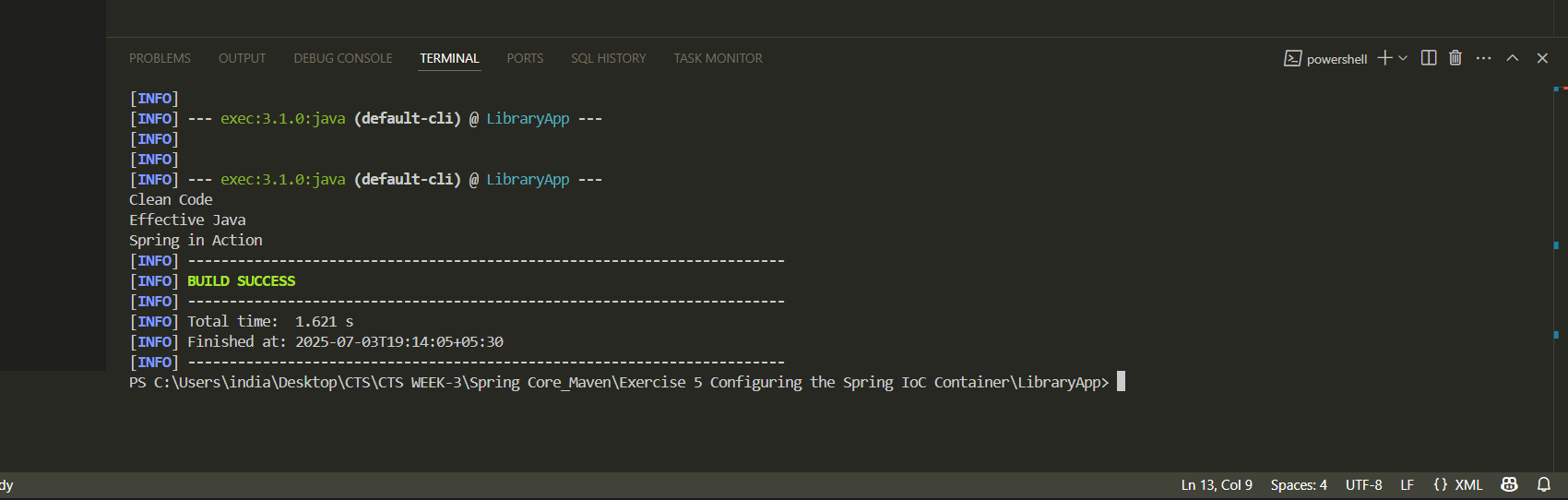
    public List<String> getAllBooks() {

        return bookRepository.findAll();

    }

}

Create a main class to load the Spring context and test the configuration



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

Update applicationContext.**xml** to configure constructor injection for **BookService**.

<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.example.library.repository.BookRepository"/>

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

        <constructor-arg ref="bookRepository"/>

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

    </bean>

</beans>

The **BookService** class has a setter method for **BookRepository**

    private final BookRepository primaryRepository;

    private BookRepository secondaryRepository;

    public BookService(BookRepository *primaryRepository*) {

        this.primaryRepository = *primaryRepository*;

    }

    public *void* setSecondaryRepository(BookRepository *secondaryRepository*) {

        this.secondaryRepository = *secondaryRepository*;

    }

    public List<String> listPrimaryBooks() {

        return primaryRepository.findAll();

    }

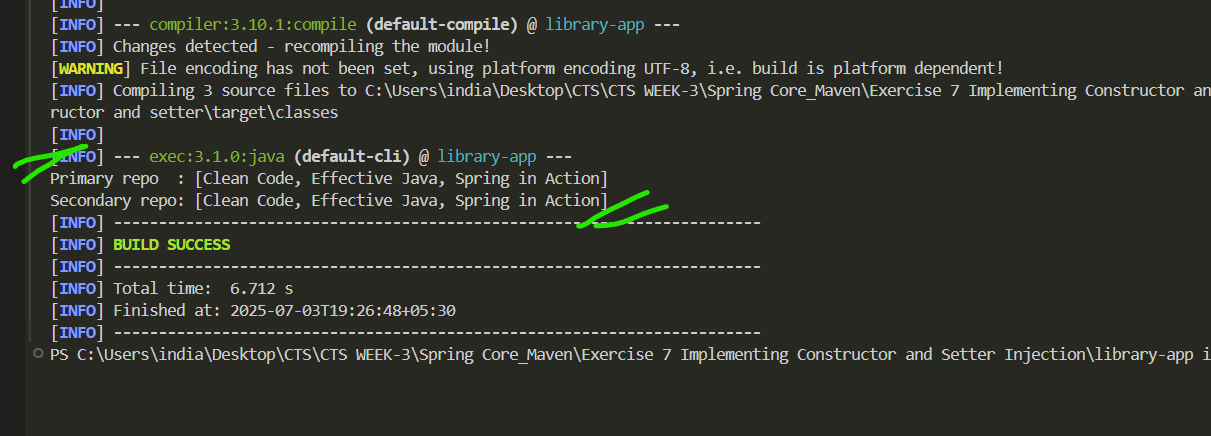
    public List<String> listSecondaryBooks() {

        return (secondaryRepository != null ? secondaryRepository : primaryRepository).findAll();

    }

}

Run the **LibraryManagementApplication**



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

Output :-

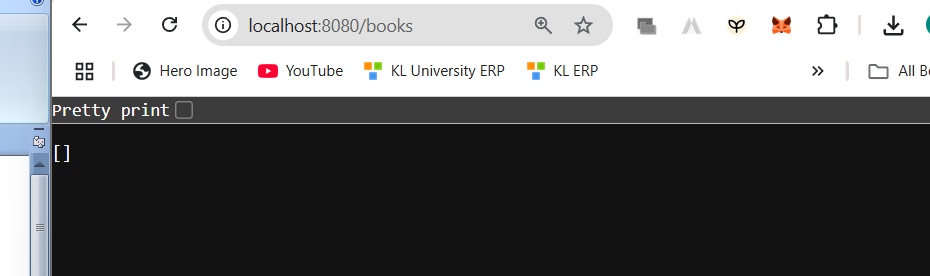
To run the application mvn spring-boot:run

we use GET POST to see the out put

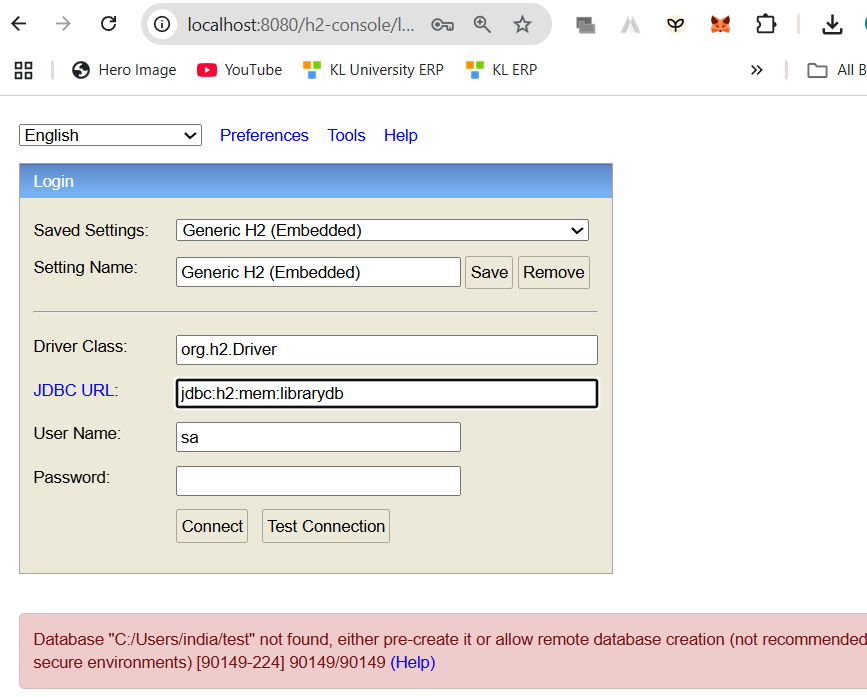
<http://localhost:8080/books>

you get this

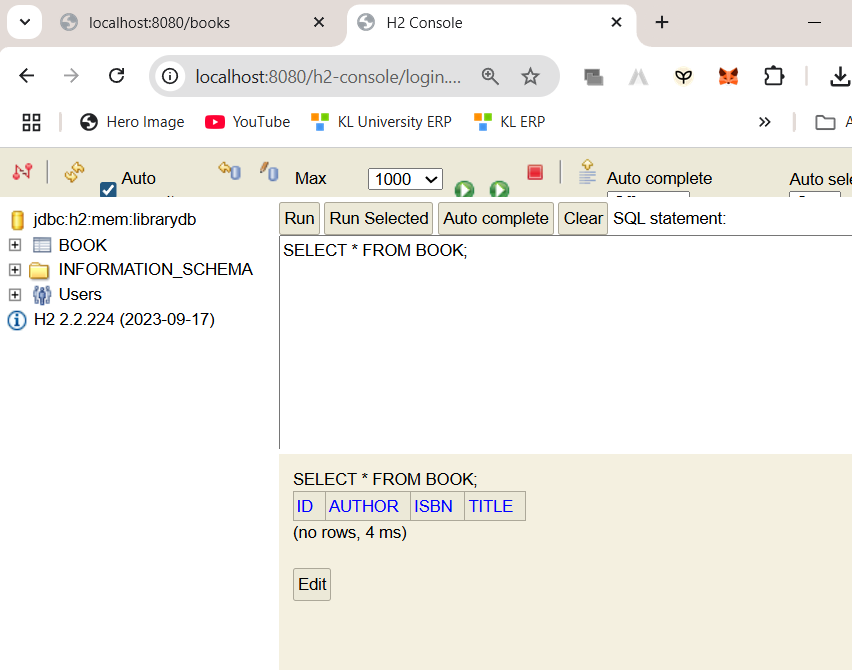
[ ] as output



If we do more changes



In the url use what you kept in application.properties.

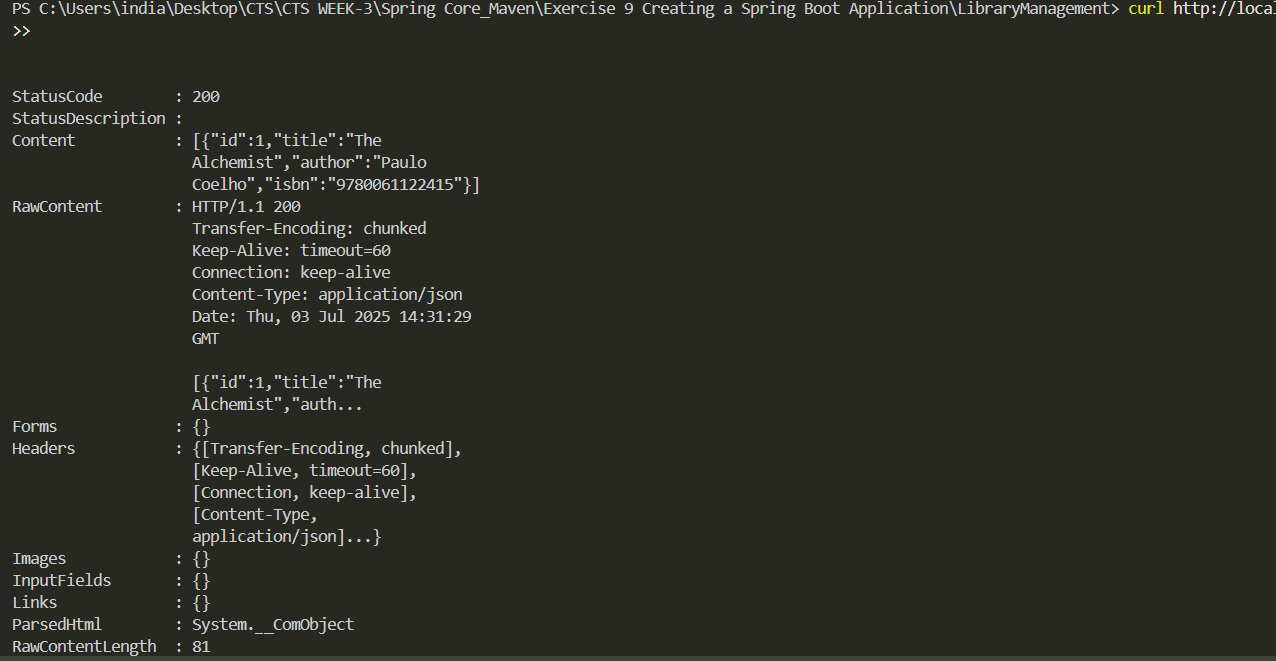


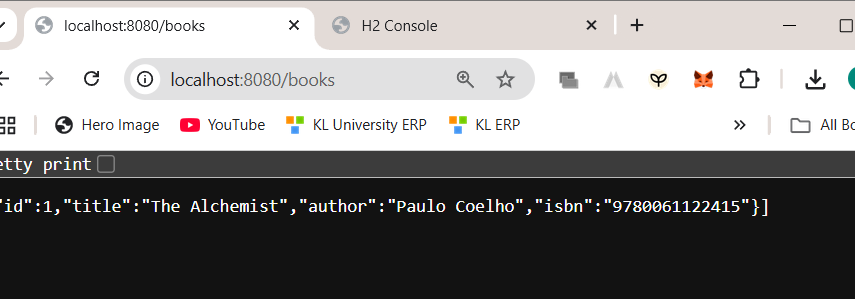
You will get to see this

curl -X POST http://localhost:8080/books -H "Content-Type: application/json" -d "{\"title\":\"The Alchemist\", \"author\":\"Paulo Coelho\", \"isbn\":\"9780061122415\"}"

try this on your cmd

you will see output in your terminal instead on POSTMAN

see the content you get the book 1 data



Dependencies used

**Spring Web, Spring Data JPA, and H2 Database**

**Create Application Properties**

# H2 in-memory DB

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

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

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

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

**Define Entities and Repositories:**

package com.example.librarymanagement.model;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@*Entity*

public class Book {

    @*Id*

    @*GeneratedValue*(strategy = GenerationType.IDENTITY)

    private *Long* id;

    private *String* title;

    private *String* author;

    private *String* isbn;

    // 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; }

    public *String* getIsbn() { return isbn; }

    public *void* setIsbn(*String* *isbn*) { this.isbn = isbn; }

}

package com.example.librarymanagement.repository;

import com.example.librarymanagement.model.Book;

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

public interface BookRepository extends *JpaRepository*<*Book*, *Long*> {

}

**Create a REST Controller:**

package com.example.librarymanagement.controller;

import com.example.librarymanagement.model.Book;

import com.example.librarymanagement.repository.BookRepository;

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

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

import java.util.List;

@*RestController*

@*RequestMapping*("/books")

public class BookController {

    @*Autowired*

    private *BookRepository* bookRepository;

    @*GetMapping*

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

        return bookRepository.findAll();

    }

    @*GetMapping*("/{id}")

    public *Book* getBookById(@*PathVariable* *Long* *id*) {

        return bookRepository.findById(id).orElse(null);

    }

    @*PostMapping*

    public *Book* createBook(@*RequestBody* *Book* *book*) {

        return bookRepository.save(book);

    }

    @*PutMapping*("/{id}")

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

*Book* book = bookRepository.findById(id).orElse(null);

        if (book != null) {

            book.setTitle(bookDetails.getTitle());

            book.setAuthor(bookDetails.getAuthor());

            book.setIsbn(bookDetails.getIsbn());

            return bookRepository.save(book);

        }

        return null;

    }

    @*DeleteMapping*("/{id}")

    public *void* deleteBook(@*PathVariable* *Long* *id*) {

        bookRepository.deleteById(id);

    }

}