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

**LibraryManagementApplication.java**

package com.library.LibraryManagement;

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);

}

}

application.properties

spring.application.name=LibraryManagement

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

**BookController.java**

package com.library.LibraryManagement.controller;

import com.library.LibraryManagement.entity.Book;

import com.library.LibraryManagement.repository.BookRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/api/books")

public class BookController {

@Autowired

private BookRepository bookRepo;

@GetMapping

public List<Book> fetchAllBooks() {

return bookRepo.findAll();

}

@GetMapping("/{id}")

public Book fetchBook(@PathVariable("id") Long bookId) {

return bookRepo.findById(bookId).orElse(new Book("Not Found", "Unknown"));

}

@PostMapping

public String registerBook(@RequestBody Book book) {

Book saved = bookRepo.save(book);

return "Book '" + saved.getTitle() + "' by " + saved.getAuthor() + " added successfully.";

}

@PutMapping("/{id}")

public String modifyBook(@PathVariable("id") Long id, @RequestBody Book newDetails) {

return bookRepo.findById(id).map(b -> {

b.setTitle(newDetails.getTitle());

b.setAuthor(newDetails.getAuthor());

bookRepo.save(b);

return "Book with ID " + id + " updated.";

}).orElse("Book not found.");

}

@DeleteMapping("/{id}")

public String removeBook(@PathVariable("id") Long id) {

if (bookRepo.existsById(id)) {

bookRepo.deleteById(id);

return "Book with ID " + id + " deleted.";

} else {

return "Book not found.";

}

}

}

**Book.java**

package com.library.LibraryManagement.entity;

import jakarta.persistence.\*;

@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 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; }

}

**BookRepository.java**

package com.library.LibraryManagement.repository;

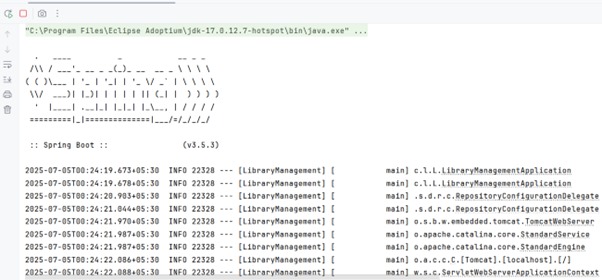
import com.library.LibraryManagement.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**Output :**

****