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

**Scenario:**

You need to create a Spring Boot application for the library management system to simplify configuration and deployment.

<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.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

</dependencies>

**Application.properties**

# DataSource configuration for H2 in-memory database

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

# JPA/Hibernate configuration

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

# Enable H2 console for database management

spring.h2.console.enabled=true

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;

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;

}

@Override

public String toString() {

return "Book{" +

"id=" + id +

", title='" + title + '\'' +

", author='" + author + '\'' +

'}';

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Book book = (Book) o;

return id != null && id.equals(book.id);

}

@Override

public int hashCode() {

return 31;

}

}

BookRepository.java

package com.library.repository;

import com.library.entity.Book;

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

import org.springframework.stereotype.Repository;

@Repository

public interface BookRepository extends JpaRepository<Book, Long> {

// Custom query methods can be added here if needed

}

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.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

@Autowired

private BookRepository bookRepository;

@GetMapping

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

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

return new ResponseEntity<>(books, HttpStatus.OK);

}

@GetMapping("/{id}")

public ResponseEntity<Book> getBookById(@PathVariable Long id) {

return bookRepository.findById(id)

.map(book -> new ResponseEntity<>(book, HttpStatus.OK))

.orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

@PostMapping

public ResponseEntity<Book> createBook(@RequestBody Book book) {

Book savedBook = bookRepository.save(book);

return new ResponseEntity<>(savedBook, HttpStatus.CREATED);

}

@PutMapping("/{id}")

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

return bookRepository.findById(id)

.map(book -> {

book.setTitle(bookDetails.getTitle());

book.setAuthor(bookDetails.getAuthor());

Book updatedBook = bookRepository.save(book);

return new ResponseEntity<>(updatedBook, HttpStatus.OK);

})

.orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

@DeleteMapping("/{id}")

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

return bookRepository.findById(id)

.map(book -> {

bookRepository.delete(book);

return new ResponseEntity<Void>(HttpStatus.NO\_CONTENT);

})

.orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

}

#### 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);

}

}