**Exercise 9: Spring Boot Application**

**1. Project Setup**

**Project Name:** LibraryManagement

**Tool:** [Spring Initializr](https://start.spring.io)

**Project Structure:**

LibraryManagement/

── src/main/java

── com.library

─ LibraryManagementApplication.java

─ controller/BookController.java

─ entity/Book.java

─ repository/BookRepository.java

── src/main/resources

── application.properties

── pom.xml

**2. Dependencies**

Add these **Maven dependencies** in pom.xml: if not used spring initializr

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Starter Data JPA -->

<dependency>

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

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

</dependency>

<!-- H2 Database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

**3. application.properties**

Add the following to **src/main/resources/application.properties** to configure the H2 database and JPA:

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.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update

**4. Define Entity and Repository**

**Book Entity**

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;

// Getters and 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; }

}

**BookRepository Interface**

package com.library.repository;

import com.library.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

// Spring Data JPA provides CRUD methods automatically

}

**5. Create REST Controller**

package com.library.controller;

import com.library.entity.Book;

import com.library.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 bookRepository;

// Get all books

@GetMapping

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

// Add a book

@PostMapping

public Book createBook(@RequestBody Book book) {

return bookRepository.save(book);

}

// Get a book by ID

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

return bookRepository.findById(id).orElseThrow(() -> new RuntimeException("Book not found"));

}

// Update a book

@PutMapping("/{id}")

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

Book book = bookRepository.findById(id).orElseThrow(() -> new RuntimeException("Book not found"));

book.setTitle(bookDetails.getTitle());

book.setAuthor(bookDetails.getAuthor());

return bookRepository.save(book);

}

// Delete a book

@DeleteMapping("/{id}")

public String deleteBook(@PathVariable Long id) {

bookRepository.deleteById(id);

return "Book deleted successfully!";

}

}

**6. Run the Application**

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);

System.out.println("Library Management Spring Boot App is running!");

}

}