**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.

**Steps:**

1. **Create a Spring Boot Project:**
   * Use **Spring Initializr** to create a new Spring Boot project named **LibraryManagement**.
2. **Add Dependencies:**
   * Include dependencies for **Spring Web, Spring Data JPA, and H2 Database**.
3. **Create Application Properties:**
   * Configure database connection properties in **application.properties**.
4. **Define Entities and Repositories:**
   * Create **Book** entity and **BookRepository** interface.
5. **Create a REST Controller:**
   * Create a **BookController** class to handle CRUD operations.
6. **Run the Application:**
   * Run the Spring Boot application and test the REST endpoints.

**CODE:**

pom.xml

<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-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

**src/main/resources/application.properties**

spring.datasource.url=jdbc:h2:mem:testdb spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect spring.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update

**src/main/java/com/example/librarymanagement/model/Book.java**

package com.example.librarymanagement.model;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

private String isbn;

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;

**src/main/java/com/example/librarymanagement/repository/BookRepository.java**

package com.example.librarymanagement.repository;

immport com.example.librarymanagement.model.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**src/main/java/com/example/librarymanagement/controller/BookController.java**

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("/api/books")

public class BookController {

private final BookRepository bookRepository;

@Autowired

public BookController(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

@PostMapping

public Book addBook(@RequestBody Book book) {

return bookRepository.save(book);

}

@GetMapping

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

@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());

book.setIsbn(bookDetails.getIsbn());

return bookRepository.save(book);

}

// Delete a book

@DeleteMapping("/{id}")

public void deleteBook(@PathVariable Long id) {

bookRepository.deleteById(id);

}

}

**OUTPUT:**

2025-07-06 18:54:09 INFO ... Starting LibraryManagementApplication

2025-07-06 18:54:10 INFO ... H2 console available at /h2-console

2025-07-06 18:54:10 INFO ... Tomcat started on port(s): 8080 (http)

2025-07-06 18:54:10 INFO ... Started LibraryManagementApplication