**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.

**LibraryManagementSystem10.java :-**

package org.example1.LibraryManagementSystem10;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class LibraryManagementApplication10 {  
 public static void main(String[] args) {  
 SpringApplication.*run*(LibraryManagementApplication10.class, args);  
 }  
}

**BookController.java :-**

package org.example1.LibraryManagementSystem10.controller;  
  
import org.example1.LibraryManagementSystem10.model.Book;  
import org.example1.LibraryManagementSystem10.repository.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
public class BookController {  
  
 @Autowired  
 private BookRepository bookRepository;  
  
 // 1. Welcome  
 @GetMapping("/")  
 public String home() {  
 return "Welcome to the Library Management System!";  
 }  
  
 // 2. Add a new book  
 @PostMapping("/books")  
 public Book addBook(@RequestBody Book book) {  
 return bookRepository.save(book);  
 }  
  
 // 3. Get all books  
 @GetMapping("/books")  
 public List<Book> getAllBooks() {  
 return bookRepository.findAll();  
 }  
  
 // 4. Get book by ID  
 @GetMapping("/books/{id}")  
 public Book getBookById(@PathVariable Long id) {  
 return bookRepository.findById(id).orElse(null);  
 }  
  
 // 5. Update book  
 @PutMapping("/books/{id}")  
 public Book updateBook(@PathVariable Long id, @RequestBody Book book) {  
 book.setId(id);  
 return bookRepository.save(book);  
 }  
  
 // 6. Delete book  
 @DeleteMapping("/books/{id}")  
 public String deleteBook(@PathVariable Long id) {  
 bookRepository.deleteById(id);  
 return "Book with ID " + id + " has been deleted.";  
 }  
  
 // 7. Search books by title (case-insensitive)  
 @GetMapping("/books/search")  
 public List<Book> searchByTitle(@RequestParam String title) {  
 return bookRepository.findByTitleContainingIgnoreCase(title);  
 }  
  
 // 8. Filter books by author  
 @GetMapping("/books/author")  
 public List<Book> filterByAuthor(@RequestParam String author) {  
 return bookRepository.findByAuthorContainingIgnoreCase(author);  
 }  
  
 // 9. Count total books  
 @GetMapping("/books/count")  
 public Long countBooks() {  
 return bookRepository.count();  
 }  
}

**Book.java :-**

package org.example1.LibraryManagementSystem10.model;

import javax.persistence.\*;

import lombok.\*;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

private String genre;

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getGenre() {

return genre;

}

public void setGenre(String genre) {

this.genre = genre;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

}

**BookRepository.java :-**

package org.example1.LibraryManagementSystem10.repository;  
  
import org.example1.LibraryManagementSystem10.model.Book;  
import org.springframework.data.jpa.repository.JpaRepository;  
import java.util.List;  
  
public interface BookRepository extends JpaRepository<Book, Long> {  
 List<Book> findByTitleContainingIgnoreCase(String title);  
 List<Book> findByAuthorContainingIgnoreCase(String author);  
}

**data.sql :-**

INSERT INTO book (title, author, genre) VALUES ('The Alchemist', 'Paulo Coelho', 'Fiction');  
INSERT INTO book (title, author, genre) VALUES ('Clean Code', 'Robert C. Martin', 'Programming');

Schema.sql :-

CREATE TABLE book (  
 id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,  
 title VARCHAR(255),  
 author VARCHAR(255),  
 genre VARCHAR(255)  
);

**Pom.xml :-**

<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
 <modelVersion>4.0.0</modelVersion>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.7.17</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <groupId>org.example1</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0.0</version>  
  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
  
 <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.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Output :-**







