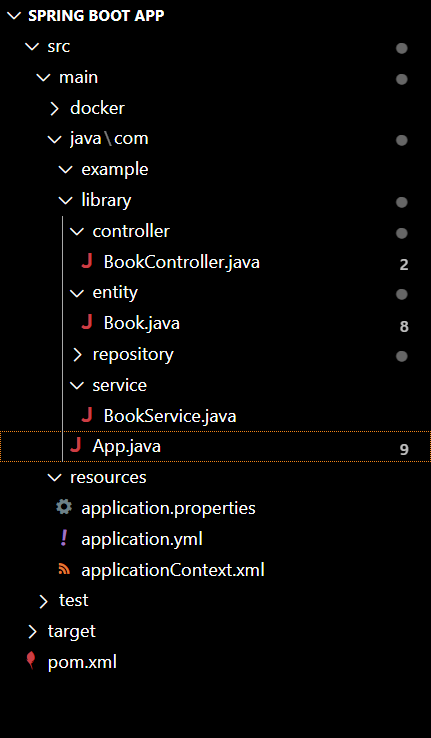
**Exercise 9: Creating a Spring Boot Application**

**Scenario:**

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

**STRUCTURE:**



**BookController.java**

package com.example.library.controller;

import com.example.library.model.Book;

import com.example.library.repository.BookRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

    @Autowired

    private BookRepository bookRepository;

    @GetMapping

    public List<Book> getAllBooks() {

        return bookRepository.findAll();

    }

    @PostMapping

    public Book addBook(@RequestBody Book book) {

        return bookRepository.save(book);

    }

}

**Book.java**

package com.example.library.model;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Book {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String title;

    private String author;

    public Book() {

        // Default constructor required by JPA

    }

    public Book(String title, String author) {

        this.title = title;

        this.author = 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.java**

package com.example.library.repository;

import com.example.library.model.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**BookService.java**

package com.example.library.service;

import org.springframework.stereotype.Service;

@Service

public class BookService {

    public String getBookInfo() {

        return "Library has Spring Boot books!";

    }

}

**App.java**

package com.example.library;

import com.example.library.model.Book;

import com.example.library.repository.BookRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class App {

    public static void main(String[] args) {

        SpringApplication.run(App.class, args);

    }

    // Optional: Insert sample books at startup

    @Bean

    CommandLineRunner initDatabase(BookRepository repository) {

        return args -> {

            repository.save(new Book("Clean Code", "Robert C. Martin"));

            repository.save(new Book("Effective Java", "Joshua Bloch"));

        };

    }

}

**applicationContext.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

       xsi:schemaLocation="

           http://www.springframework.org/schema/beans

           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- BookRepository bean -->

    <bean id="bookRepository" class="com.example.library.repository.BookRepository" />

    <!-- BookService bean with constructor and setter injection -->

    <bean id="bookService" class="com.example.library.service.BookService">

        <!-- Constructor injection -->

        <constructor-arg value="Library Book Service" />

        <!-- Setter injection -->

        <property name="bookRepository" ref="bookRepository" />

    </bean>

</beans>

**Application.properties**

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.show-sql=true

**OUTPUT:**

