Spring Boot Library Management System Documentation

Name: Pawar Rohit S.

Roll No.: 115

1. Introduction

This documentation provides a detailed guide for creating a Library Management System web application using Spring Boot. The purpose of this application is to manage library resources, including books, members, and transactions, with a user-friendly interface and RESTful APIs.

2. Objectives

- To create a library management application using Spring Boot.

- To understand the structure of a Spring Boot project in a real-world scenario.

- To implement CRUD operations for books and members.

- To build and run RESTful web services for library transactions.

- To learn how to configure persistence and test the application.

3. Technologies Used

- Spring Boot

- Java 11 or higher

- Maven

- Spring Web

- Spring Data JPA

- H2/MySQL Database

- IDE: IntelliJ IDEA / Eclipse

4. Project Structure

LibraryManagementSystem/

├── src/

```
│   ├── main/
│   │   ├── java/com/example/library/
│   │   │   ├── LibraryManagementSystemApplication.java
│   │   │   ├── controller/
│   │   │   │   ├── BookController.java
│   │   │   │   ├── MemberController.java
│   │   │   │   └── TransactionController.java
│   │   │   ├── model/
│   │   │   │   ├── Book.java
│   │   │   │   ├── Member.java
│   │   │   │   └── Transaction.java
│   │   │   └── repository/
│   │   │       ├── BookRepository.java
│   │   │       ├── MemberRepository.java
│   │   │       └── TransactionRepository.java
│   └── resources/
│       └── application.properties
├── pom.xml
```

5. Main Application Class (LibraryManagementSystemApplication.java)

```
package com.example.library;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class LibraryManagementSystemApplication {
    public static void main(String[] args) {
        SpringApplication.run(LibraryManagementSystemApplication.class, args);
    }
```

}

6. Controller Classes

BookController.java

```
@GetMapping("/books")
   public List<Book> getAllBooks() { ... }

@PostMapping("/books")
   public Book addBook(@RequestBody Book book) { ... }
```

MemberController.java

```
@GetMapping("/members")
   public List<Member> getAllMembers() { ... }

@PostMapping("/members")
   public Member addMember(@RequestBody Member member) { ... }
```

TransactionController.java

```
@PostMapping("/borrow")
   public Transaction borrowBook(@RequestParam Long bookId, @RequestParam Long
memberId) { ... }

@PostMapping("/return")
   public Transaction returnBook(@RequestParam Long transactionId) { ... }
```

7. application.properties

```
# Database configuration
spring.datasource.url=jdbc:h2:mem:librarydb
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
```

8. Sample 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
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.example</groupId>
```

```
    <artifactId>LibraryManagementSystem</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <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>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

9. How to Run the Application

1. Open the project in your preferred IDE (IntelliJ IDEA, Eclipse).

2. Build the project using Maven.

3. Run the `LibraryManagementSystemApplication.java` main class.

4. Open a web browser and go to `http://localhost:8080/books` to view all books.

5. Use API testing tools (e.g., Postman) to test endpoints for adding books, members, and transactions.

10. Conclusion

This Library Management System application serves as a comprehensive example to learn Spring Boot. It demonstrates how to configure persistence, implement RESTful endpoints, and manage basic library operations.

11. References

1. https://spring.io/guides/gs/spring-boot/

2. https://www.baeldung.com/spring-boot-start

3. https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

4. https://www.geeksforgeeks.org/spring-boot-restful-web-services/

5. https://www.javatpoint.com/spring-boot-crud-rest-api