Library Management System Documentation

Table of contents

Project Overview

Project Name: Maids.cc Library Management System

Description

The Maids.cc Library Management System is a robust API built using Spring Boot. It facilitates efficient management of books, patrons, and borrowing records in a library setting. Librarians can easily handle book additions, updates, and removals, as well as patron management and borrowing transactions.

Entities

Book: Represents information about each book in the library, including attributes such as ID, title, author, publication year, ISBN, etc. Patron: Contains details about library patrons, including ID, name, contact information, etc. Borrowing Record: Tracks the association between books and patrons, including borrowing and return dates.

API Endpoints

Book Management Endpoints:

1. GET /api/books: Retrieve a list of all books.
2. GET /api/books/: Retrieve details of a specific book by ID.
3. POST /api/books: Add a new book to the library.
4. PUT /api/books/: Update an existing book's information.
5. DELETE /api/books/: Remove a book from the library.

Patron Management Endpoints:

1. GET /api/patrons: Retrieve a list of all patrons.
2. GET /api/patrons/: Retrieve details of a specific patron by ID.
3. POST /api/patrons: Add a new patron to the system.

4. PUT /api/patrons/: Update an existing patron's information.

5. DELETE /api/patrons/: Remove a patron from the system.

Borrowing Endpoints:

1. POST /api/borrow//patron/: Allow a patron to borrow a book.

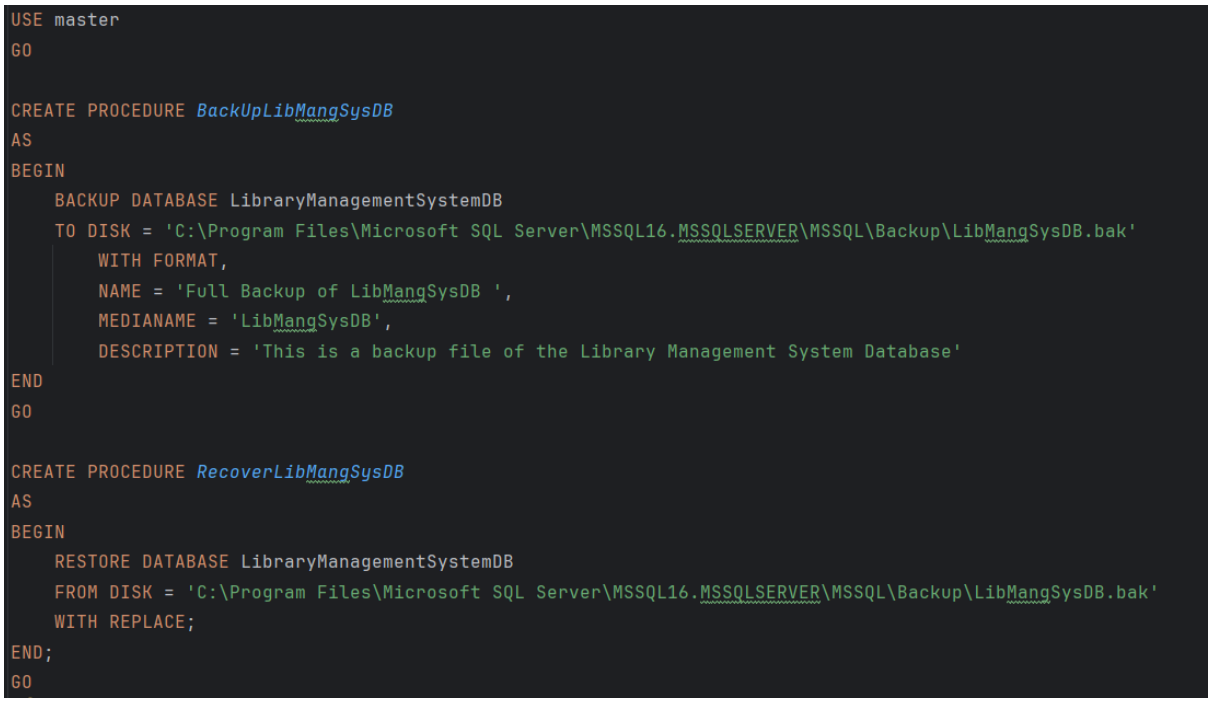
2. PUT /api/return//patron/: Record the return of a borrowed book by a patron.

Database

The library management system is designed to streamline the organization and retrieval of library resources. At the heart of this system lies a robust database schema, implemented using Microsoft SQL Server, which ensures efficient data management and integrity. The schema is composed of three primary entities: PATRON, BOOK, and BORROWING. The PATRON entity captures essential information about library members, including a unique ID, personal details, and contact information. The BOOK entity records the details of the library's book inventory, such as title, author, and publication year, each identified by a unique ID. The BORROWING entity serves as the linchpin that connects patrons to the books they borrow, tracking the borrowing period with start and end dates. This relational schema facilitates a one-to-many relationship between patrons and books, allowing for multiple borrowings while maintaining a clear record of transactions. By leveraging the capabilities of SQL Server, the system provides a scalable and reliable foundation for managing the library's operations.

Backup And Recovery

The backup and recovery aspect of the DB was implemented to allow the DBA the opportunity to backup the current state of the DB and copy it to any external secondary storage medium.



```
USE master
GO

CREATE PROCEDURE BackUpLibMangSysDB
AS
BEGIN
    BACKUP DATABASE LibraryManagementSystemDB
    TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\LibMangSysDB.bak'
        WITH FORMAT,
        NAME = 'Full Backup of LibMangSysDB ',
        MEDIANAME = 'LibMangSysDB',
        DESCRIPTION = 'This is a backup file of the Library Management System Database'
END
GO

CREATE PROCEDURE RecoverLibMangSysDB
AS
BEGIN
    RESTORE DATABASE LibraryManagementSystemDB
    FROM DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\LibMangSysDB.bak'
    WITH REPLACE;
END;
GO
```

backup_recovery.png

Borrowing Trigger

A Trigger has been developed to insure the DB consistency when adding a borrow record, that is, we need to ensure the book being borrowed is not currently borrowed by someone else.

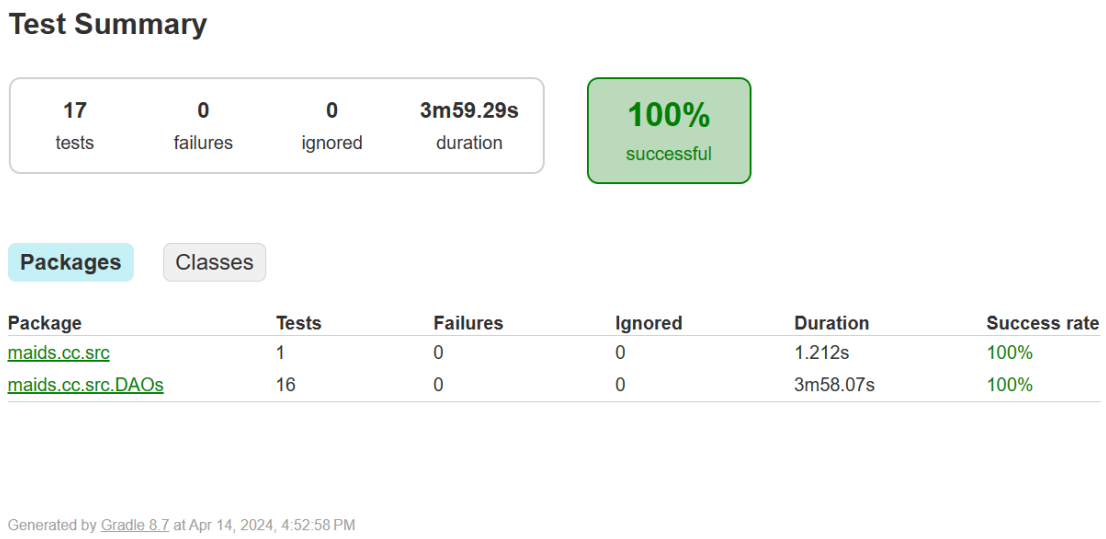
```
CREATE TRIGGER CheckBookAvailability
ON BORROWING
INSTEAD OF INSERT
AS
BEGIN
  SET NOCOUNT ON;
  DECLARE @BookID INT;
  SELECT @BookID = book_id FROM INSERTED;

  IF EXISTS (
    SELECT 1
    FROM BORROWING b WITH (UPDLOCK, HOLDLOCK)
    WHERE b.book_id = @BookID AND b.end_date = null
  )
  BEGIN
    RAISERROR ('Book with ID %d is currently borrowed and unavailable.', 16, 1, @BookID);
  END;
  ELSE
  BEGIN
    INSERT INTO BORROWING (patron_id, book_id, start_date, end_date)
    SELECT patron_id, book_id, start_date, end_date FROM INSERTED;
  END;
END;
```

borrow_trigger.png

Testing

Due to the short time window given, unfortunately I was not able to cover multiple classes. However, I was able to cover only 2 of the 3 DAOs implemented, and no Endpoints were tested professionally.



Test Summary

17 tests	0 failures	0 ignored	3m59.29s duration	100% successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
maids.cc.src	1	0	0	1.212s	100%
maids.cc.src.DAOs	16	0	0	3m58.07s	100%

Generated by Gradle 8.7 at Apr 14, 2024, 4:52:58 PM

testing.png

Backend Server

The class diagram is attached in the `Design` directory within the project repository.

Helpful Resources

This section of the software documentation, I'll be attaching few online free resources that I've found helpful in some point during the software development.

1. Logging With AOP in Spring (https://www.baeldung.com/spring-aspect-oriented-programming-logging)
2. Implementing Logging Using Aspect-Oriented Programming (AOP) - Spring: .. (https://youtu.be/fdRKqzEWJqw?si=fdQthyFrDDmzvbPf)