Book Management API

This project is a .Net Core MVC REST API for managing a collection of books. It includes various functionalities for sorting, retrieving, and saving book data, as well as generating citations in MLA and Chicago styles. The API interacts with a SQL Server/SQLite database to store book information.

Repository Pattern with Dapper and Dependency Injection

Overview

This project implements the repository pattern to abstract and manage data access logic using Dapper, a lightweight ORM. The repository pattern ensures a clean separation of concerns, making the code more modular and testable.

Implementation Details

- \*Repository Pattern\*: The repository pattern is used to encapsulate data access logic. Interfaces define the contract for data operations, while concrete repository classes implement these operations using Dapper.

The application follows a layered architecture with a service class (**BookService.cs**) and a repository class (**BookRepo.cs**). The API constructor accesses the service methods to perform operations.

The application uses SQLite as an in-memory database and SQL Server for persistent storage. The connection strings for both databases are defined in the **appsettings.json** file and retrieved in the **DbContext.cs** class.

- \*Dependency Injection\*: The repositories are registered in the IoC container with a scoped lifetime, ensuring that a single instance is used within the scope of a request. This is configured in the **Startup.cs** file, promoting efficient resource management and better application performance.

Usage

- \*API Controllers\*: Repositories are injected into API controllers via constructor injection. This adheres to dependency inversion principles, enhancing testability and maintainability. Controllers interact with the repository to perform CRUD operations.

By leveraging the repository pattern and dependency injection, the application maintains a clear separation between business logic and data access logic, resulting in a more maintainable and scalable codebase.

Table of Contents

1. [API Methods](#api-methods)

2. [Database Design](#database-design)

3. [Stored Procedures](#stored-procedures)

4. [Book Class](#book-class)

API Methods

Get Sorted List by Publisher, Author, Title

- \*\*Endpoint\*\*: `/api/books/sorted-by-publisher-author-title`

- \*\*Method\*\*: `GET`

- \*\*Description\*\*: Returns a list of books sorted by Publisher, Author (last, first), and Title.

Get Sorted List by Author, Title

- \*\*Endpoint\*\*: `/api/books/sorted-by-author-title`

- \*\*Method\*\*: `GET`

- \*\*Description\*\*: Returns a list of books sorted by Author (last, first) and Title.

Get Total Price of All Books

- \*\*Endpoint\*\*: `/api/books/total-price`

- \*\*Method\*\*: `GET`

- \*\*Description\*\*: Returns the total price of all books in the SqLite database.

Save List of Books

- \*\*Endpoint\*\*: `/api/books/save`

- \*\*Method\*\*: `POST`

- \*\*Description\*\*: Saves a list of books to the SqLite database in a single call.

Bulk Insert for SqlServer

- \*\*Endpoint\*\*: `/api/books/bulkInsertBooks `

- \*\*Method\*\*: `GET`

- \*\*Description\*\*:Bulk data transfer to SqLite to SqlServer and logged on history.

Get Book detail for SQL server

- \*\*Endpoint\*\*: `/api/books/getall-books-detail`

- \*\*Method\*\*: `GET`

- \*\*Description\*\*:Return the all book details on SQL server.

Generate Citation style as MLA Styles

- \*\*Endpoint\*\*: `/api/books/style-citation`

- \*\*Method\*\*: `Get'

- \*\*Description\*\*: Return the citation style of the user book as mla.

Generate Citation style as Chicago

- \*\*Endpoint\*\*: `/api/books/style-chicago `

- \*\*Method\*\*: `Get'

- \*\*Description\*\*: Return the citation style of the user book as Chicago.

Database Design

Table: Books

- \*\*Id\*\*: `int` (Primary Key, Identity)

- \*\*Title\*\*: `nvarchar(255)`

- \*\*AuthorFirstName\*\*: `nvarchar(100)`

- \*\*AuthorLastName\*\*: `nvarchar(100)`

- \*\*Publisher\*\*: `nvarchar(255)`

- \*\*Price\*\*: `decimal(18, 2)`

Stored Procedures

Get Sorted List by Publisher, Author, Title

```sql

CREATE PROCEDURE GetBooksSortedByPublisherAuthorTitle

AS

BEGIN

SELECT \* FROM Books

ORDER BY Publisher, AuthorLastName, AuthorFirstName, Title;

END

```

Get Sorted List by Author, Title

```sql

CREATE PROCEDURE GetBooksSortedByAuthorTitle

AS

BEGIN

SELECT \* FROM Books

ORDER BY AuthorLastName, AuthorFirstName, Title;

END

```

Get Total Price of All Books

```sql

CREATE PROCEDURE GetTotalPrice

AS

BEGIN

SELECT SUM(Price) AS TotalPrice FROM Books;

END

```

Book Class

Properties

- `Id` : `int`

- `Title` : `string`

- `AuthorFirstName` : `string`

- `AuthorLastName` : `string`

- `Publisher` : `string`

- `Price` : `decimal`

- `Role ` : `role`

Enum

- `Admin` : `1`

- `User` : `2 `

MlaCitationBookModel

Properties

- `AuthorFirstName` : `string`

- `AuthorLastName` : `string`

- `Publisher` : `string`

- `TitleOfSource` : `string`

- `TitleOfContainer` : `string`

- `PublicationDate ` : `string`

- `Location` : `string`

ChicagoBookModel

Properties

- `AuthorFirstName` : `string`

- `AuthorLastName` : `string`

- ` PublicationDate ` : `string`

- `TitleOfSource` : `string`

- `JournalTitle` : `string`

- `VolumeNo ` : `string`

- `IssueNo` : `string`

- `PageRange` : `string`

- `URL` : `string`

- `MLACitation` : `string`

- \*\*Description\*\*: Outputs the book's citation in MLA style.

- `ChicagoCitation` : `string`

- \*\*Description\*\*: Outputs the book's citation in Chicago style.

Citation Generation

The `MLACitation` and `ChicagoCitation` properties generate citations based on the book's details, formatted according to the respective styles.

Summary

This project demonstrates how to create a REST API using .Net Core MVC to manage a collection of books with functionalities for sorting, retrieving, and saving book data, as well as generating citations. The API integrates with a SQL Server/SQLite database and uses stored procedures for efficient data operations. The `Book` class includes properties for generating MLA and Chicago style citations.