**BookSearchApp Developer Documentation**

**Introduction:**

This software is used to search for books from the Open Library.

Users can search for relevant content according to book titles and author names.

Users can click on the search results to view detailed information, or click on the author's name to view the author's profile.

**Description of architecture:**

**View:** The user interface of the application, including various pages, controls, and view models. The View layer is written in XAML and C# and communicates with the ViewModel layer using data binding.

**ViewModel:** This layer handles the application logic and connects the View and Model layers. It contains code that interacts with the user interface, such as commands, events, and properties, and is also responsible for passing data obtained from the Model layer to the View layer. The ViewModel layer is written in C# and implements the INotifyPropertyChanged interface to notify the View layer of data changes.

**Model:** This layer handles the data operations of the application, including data storage, data retrieval, and data processing, etc. The Model layer is written in C# language, and implements data access objects, entity classes and returns the results to the ViewModel layer.

**Service:** The Service layer provides services for accessing remote data sources. Separated from the Model layer, it is responsible for interacting with the data source (https://openlibrary.org/developers/api) to improve the performance and response speed of the application. The Service layer is written in C# language and implements data access objects, HTTP clients, etc.

**Introduction of main components:**

**Main page (MainPage):**

**Search Box (TextBox):** This control is used to receive the search keywords entered by the user and pass them to the ViewModel layer through event binding to search for related books.

**Grid View (GridView):** This control is used to display a list of books, and each list item includes information such as the book cover, title, author, and publication year. When the user clicks on the list item, the information of the selected book is passed to the ViewModel layer through event binding, so that the detailed information of the book can be displayed. Or when the user clicks on the author's name, the information of the selected book is passed to the ViewModel layer through event binding to display the author's profile information. The picture of the book can also be clicked to make it zoomable.

**BookDetailPage:** This page is used to display the detailed information of the selected book, including the book cover, title, author and brief introduction. When the user returns to the list page, the page navigation operation is realized through the Frame control.

**Author profile information (AuthorPage):** This page is used to display the profile information of the selected author, including author name, other name, year of birth, author profile and author wiki. When the user returns to the list page, the page navigation operation is realized through the Frame control.

**List of classes:**

**Model class:**

**AuthorInformation.cs:** used to store Data on Individual Authors

**BooksResponse.cs:** used to store Response Format, and the format of each book in the book list in Response Format

**ModelViews class:**

**AuthorPageViewModel.cs:** Pass the author data obtained by the API in the Service class from the Model layer to the View layer.

**BookDetailPageViewModel.cs:** Pass the book data obtained from the Model layer by the API in the Service class to the View layer.

**MainPageViewModel.cs:** Pass the book list data obtained by the api in the Service class from the Model layer to the View layer.

**Views class:**

**AuthorPage.xaml/cs:** Accept the author key data transmitted by MainPage, get the data transmitted by api and display the author information on the new page

**BookDetailPage.xaml/cs:** Accept the book data transmitted by MainPage and display it on a new page

**MainPage.xaml/cs:** Display the search box, return the user input data to the ModelViews layer and accept the data transmitted by the api to be displayed in the GridView

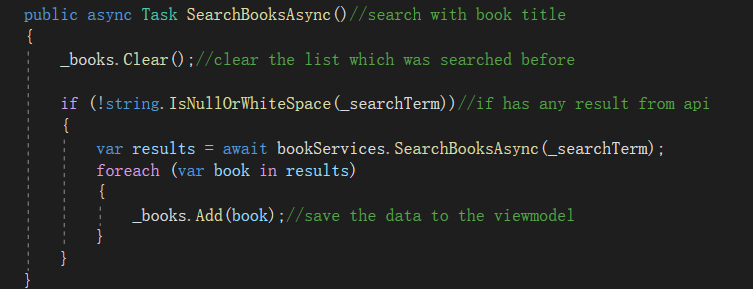
**Services class:**

**BookServices.cs:** Provides services for accessing remote data sources, stores all required api calls to improve application performance and response speed, and waits for ModelViews layer calls to obtain remote data

**Demonstration of the complete process of searching by book title:**

1. The user enters the book title in the search box and clicks the "Search" button.

2. The ModelViews layer obtains the text in the search box, and uses the text as a parameter to initiate an HTTP request using the method of BookServices. HTTP requests are implemented using the HttpClient class.

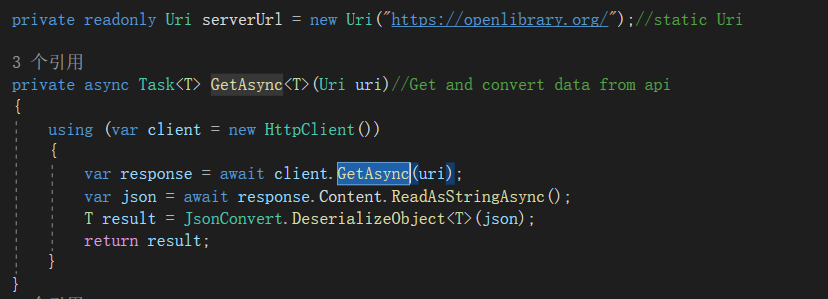


3. The application sends an HTTP request to the Web API, which contains the following information:

HTTP method: GetAsync method.

Request URL: The interface address of the Web API for searching books.

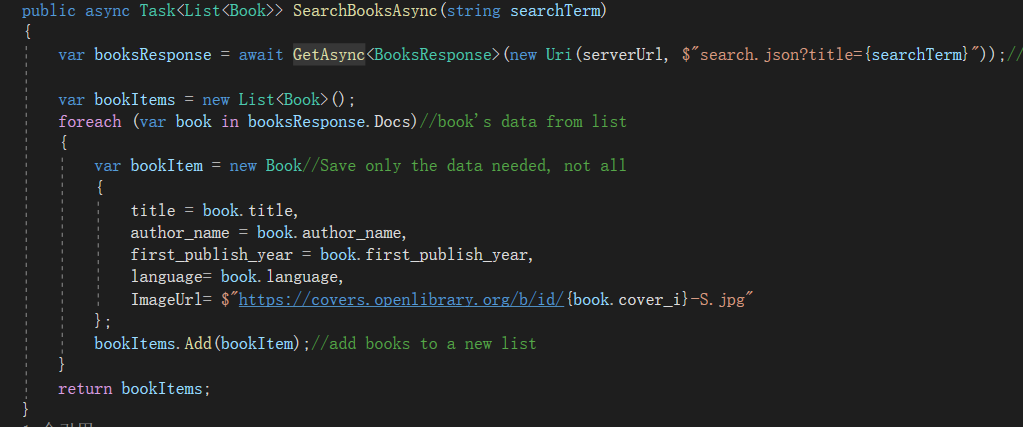
Request Parameters: Send the text in the search box as query parameters.



4. After the Web API receives the HTTP request, it obtains the book information that meets the query conditions from the database.

5. The Web API encapsulates the queried book information into JSON format and returns it to the application as an HTTP response.

6. After the application receives the HTTP response, it uses the JSON serialization tool to convert the response data into a local list of Book objects.



7. The application binds the list of Book objects to a UI control (GridView) so that the user can view the search results.