**About the BookFinder API**

The BookFinder is a REST API that allows you to search for and retrieve information about books. The API does not require authentication to make requests and it allows you to find and retrieve, or create a variety of different resources including the following: title, author, genre and rating.

Using the BookFinder API offers several benefits for people looking to incorporate book-related data into their applications. Here are a few key benefits:

* Users can quickly and easily integrate book-related data into their applications. This saves time and effort compared to trying to scrape book data from various sources.
* The BookFinder API provides comprehensive data on books. This makes it a valuable resource for building book-related applications.
* It allows for customized queries based on a variety of parameters, including book title, author, genre, and more. This allows users to tailor their requests to specific needs and retrieve the most relevant data.

Overall, using the BookFinder API can provide a convenient and efficient way to access book data and can help you save time and create more useful and engaging book-related applications. The target users are, or could be, web developers, software students, researchers and business owners.

**Getting Started**

1. Familiarize yourself with this API documentation. This will give you an overview of the API, its endpoints, parameters, and response formats.
2. Test the API: Before integrating the API into your application, it's a good idea to test it using a tool like Postman. This will allow you to send requests to the API and see the responses it returns.
3. Integrate the API into your application: Once you have tested the API and are familiar with how it works, you can start integrating it into your application.

**How it works**

The BookFinder API can be used to search for books by title, author, genre and rating. You can use HTTP requests to send queries to the API and receive responses in JSON format. To start, you will need to make an HTTP request to the API endpoint, which is typically a URL that starts with "https://" followed by the domain name of the API provider and the path to the specific API resource you want to access.

**Endpoints**

An endpoint is a specific URL within an API that can be accessed by you to perform specific actions. Endpoints provide a way for you to interact with the resources exposed by the API. The endpoints for the BookFinder API are:

’GET /books’

*This endpoint returns a list of books.*

‘GET /books/:id’  
*This endpoint is used to retrieve a specific book by its id.*

’POST /books’  
*This endpoint is used to add a new book to the database.*

’PUT /books/:id’  
*This endpoint updates an existing book with the specified id.*

’DELETE /books/:id’  
*This endpoint is used to delete a specific book by its id.*

**Query Parameters**

Query parameters allow you to control what data is returned from a request. For example, a query parameter may let you specify how many items are returned in the response. A query parameter is appended to the end of a URL after a question mark (?) and is used to filter, sort, and paginate data that is returned by an API endpoint.

**The BookFinder’s Query Parameters:**

In this API, query parameters are used to filter and sort the books collection based on specific criteria. The available query parameters include:

’title’ - the title of the book to search for

’author’ - the author of the book to search for

’genre’ - the genre of the book to search for

’rating’ – the rating of the book to search for

’limit’ – the chosen number of books to return

‘page’ – page size, i.e. the number of books to return on one page

‘sort’ – sort published dates in ascending (default) or descending order (-1)

When any of the endpoints are called with a query parameter, the API will return what is specified in the request. Under the next heading you will see how the query parameters can be used in the BookFinder API to provide flexible and customized responses:

**Examples with endpoints and query parameters**

The below examples include{{baseURL}}, which, in this case, is a variable in Postman that is used to represent the base URL of the API. For example, if your API is hosted on ‘https://bookfinder.com/api’, then you can set {{baseURL}} to ‘https://bookfinder.com’ in your Postman collection or environment. By using this variable instead of hard-coding the base URL in your requests, you can easily switch between different environments without having to update each request individually.

1. If you want to retrieve information on a specific title, the endpoint URL could be:

*{{baseURL}}/api/books?title=Klara and the Sun*

1. If you want to access one particular author, the endpoint URL could be:

*{{baseURL}}/api/books?author=Kazuo Ishiguro*

1. If you want to access all books in the genre ‘Fantasy’, the API resource you want to access is:

*{{baseURL}}/api/books?genre=Fantasy*

The endpoint in this case is ‘/api/books’ with a query parameter ‘genre=Fantasy’. The API server will receive this request, interpret the query parameter ‘genre’ with a value of ‘Fantasy’ and return a response with a list of books that match the specified genre.

1. If you want to retrieve all books that have a rating of 4, the endpoint URL is:

*{{baseURL}}/api/books?rating=4*

Any integer between 1 and 5 can be used in this API.

1. If you want to retrieve the first three books from the database, the API resource you want to access is:

*{{baseURL}}/api/books?limit=3*

1. To retrieve the first five books in the collection, your request would be:

*{{baseURL}}/api/books?limit=5&page=1*

This endpoint is used to retrieve a paginated list of books. The ‘limit’ parameter is used to specify the maximum number of books to be returned per page, while the ‘page’ parameter is used to specify the page number of the results to be returned.

1. If you want to retrieve the books sorted by published date in descending order, the API resource you want to access is:

*{{baseURL}}/api/books?sort=publishedDate-1*

**A step-by-step guide for making a request**  
This step-by-step guide will walk you through the process of using the BookFinder API to search for books, add new books to the database, update information about a book, and even delete a book. It is an advantage if you have Postman installed on your computer, which is a popular tool for testing and debugging APIs.

To make a request, first find the HTTP method and the path for the operation that you want to use. The HTTP methods are: GET, POST, PUT and DELETE.

1. **The GET method**

Once you have the API endpoint URL, you can use the HTTP GET method to retrieve data from the API. As explained above you can include parameters in your HTTP request to specify search criteria, such as the title, author, genre, rating, or limit of the book(s) you are looking for.

1. Determine the base URL for the API endpoint you want to access.

Append any necessary parameters to the URL in the format of ‘?parameter=value’. Example of GET-request that will retrieve all books in the database: [*{{baseURL}}//api/books*](https://Bookfinder.com/api/books)

1. Send the request by pressing the send button and wait for the response from the server.

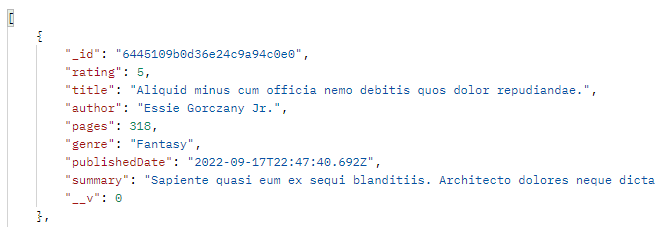
**Response for the GET-method**

When you send a GET request to a server, the server processes the request and generates a response in JSON format that includes information about the books that match your search criteria.

The response typically includes a status code, headers, and a message body. The status code indicates whether the request was successful or not. If you get the status code 200 OK, it means the request was successful (more about codes under the heading ‘Error handling’ further down).

The message body contains the actual data that the server is sending back. For example, if you requested information about all books, the message body of the response includes details about all books.

This print screen shows the first book object in the response from a GET request:



*Picture 1: Response (the message body) from a GET request, the first object.*

A book object has the following properties:

’id’ - the unique ID of the book

’rating’ – the rating of the book

’title’ - the title of the book

’author’ - the author’s first and last name

’genre’ - the genre of the book

’pages’ - the number of pages of the book

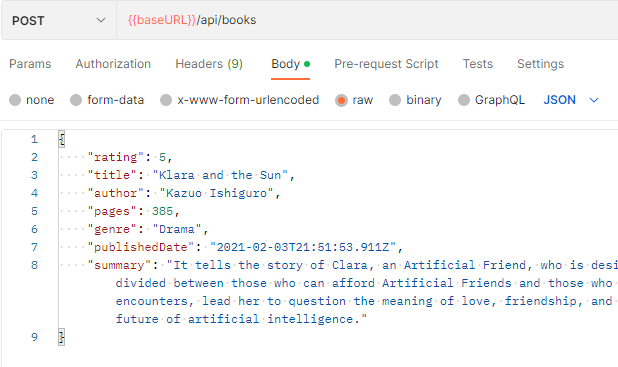
’publishedDate’ - the date the book was published

’summary’ – a brief description of the book’s plot

1. **The POST method**

The POST method adds a new book to the database with the information you submit.

1. Decide what information you want to submit to the API to create a new book.
2. Open your preferred HTTP client (e.g. Postman) and create a new POST request to the endpoint: [*{{baseURL}}//api/books*](https://Bookfinder.com/api/books)
3. The following fields can/must be entered into the ‘Request Body’*.*
   * title (required)
   * author (required)
   * pages
   * genre
   * publishedDate (required)
   * summary
   * rating
4. In the ‘Request Body’, choose radio button ‘raw’ and include a JSON object with the information you want to submit. Here is an example of a JSON object:



*Picture 2: The request body in a POST request.*

1. Send the request and wait for the API to respond.

**Response for the POST-method**

If the request is successful, the API will return a response with a 201 status code and the JSON object of the newly created book, including its unique id. Please check the response received to ensure that the book was created with correct information.

1. **The PUT method**

This method updates an existing book with the specified id.

1. Decide what information you want to submit to the API to update a book.
2. Open your preferred HTTP client (e.g. Postman) and create a new PUT request endpoint for the book you want to update, including the book's unique identifier:[*{{baseURL}}/api/books/{id}*](http://localhost:3480/api/books/%7bid%7d)
3. Add the request body in JSON format, containing only the information you want to update.
4. Send the request to the API server by clicking on the Send button.

**Response for the PUT-method**

The server will receive the request and update the book's information in the database if the book is found. The server will return a response with the updated book information in JSON format and a 200 status code if the update was successful.

If the book is not found, the server will return a 404 status code with an error message. Do not forget to verify the response received to ensure that the book was updated correctly.

1. **The DELETE method**

This method deletes the book with the specified id.

1. Identify the ID of the book you want to delete. You can retrieve this by making a GET request to the API and looking at the ID field of the book you want to delete.
2. Construct the endpoint URL for the book you want to delete. The endpoint should be in the format [*{{baseURL}}/api/books/{id}*](http://localhost:3480/api/books/%7bid%7d) where {id} is the id of the book you want to delete.
3. Send the request

**Response for the DELETE method**

Check the response from the server to make sure the book was successfully deleted. If the book was deleted successfully, the server should respond with a 200 status code (indicating that the request was successful). If there was an error, the server may respond with a 404 status code (indicating that the book was not found) or a 500 status code (indicating a server error).

If the book was successfully deleted, you can confirm this by making another get request to the API and checking that the book is no longer returned in the response.

**Error handling**

BookFinder API uses conventional HTTP response codes to indicate the success or failure of an API request.

* Codes in the 2xx range indicate success.
* Codes in the 4xx range indicate an error that failed given the information provided.
* Codes in the 5xx range indicate an error with the server

The BookFinder API returns the following error codes:

400 - Bad Request

404 - Not Found

500 - Server Error

**API limitations**

The BookFinder APIs has a set of specific functions, and it is not possible to create an API that can perform every function. For example, you cannot search specifically for a summary, a certain publish date or a certain amount of pages as this is not prioritized within the timeframe of this project. The API handles only the most common errors. The sort function only works for published dates.

**Conclusion**

Thank you for using the BookFinder API. If you have any questions or feedback, please contact the team at [support@Bookfinder.com](mailto:support@Bookfinder.com)