

Scenario:

Imagine you're a Python developer working for a small library. Your job is to create an API that allows users to:

- See all the books in the library.
- Search for books using the title or author's name.
- Add new books to the library (only authorized users can do this).

You'll use Flask-RESTful to build this API. We'll also cover some basic security measures to protect the API and how to handle errors gracefully.

Objective:

You're going to build a RESTful API for a library. This API will let people interact with the library's book catalog. Think of it like building a digital librarian that can understand and respond to specific requests.

Instructions:

Simplified and Streamlined Instructions

Step 1: Set Up the API Framework (5 minutes)

Install Flask-RESTful: In Jupyter notebook, run:

```
!pip install flask-restful
```

Configure Flask-RESTful: Create a new Python file (e.g., library_api.py) and add the following code to set up the framework:

```
4
```

Create an Endpoint to List All Books (GET /books/)

Add this class to list all books:

```
1
2
3
4
5
6
7
8
9
10
11
```

Add a Search Functionality (GET /books/?search=<query>)

Modify the get method to allow searching:

```
1
2
3
4
5
6
7
~
```

8
9
10
11

Add an Endpoint to Create a New Book (POST /books/)

Use this method to handle adding new books:

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Handle Errors (404 Not Found)

Add a separate resource to handle requests for specific book IDs:

1
2
3
4
5

Link the resources to their respective routes:

2

Run the Flask application:

1
2
3
4
5

5
6
7

Testing the API

- Use `/books/` to list all books.
- Add a search query (e.g., `/books/?search=1984`) to search by title or author.
- Test adding new books with a POST request and authentication credentials (`admin:password`).
- Test error handling by requesting a non-existent book (`/books/999`) or submitting invalid data.

Now that you've built your API, it's time to test it and make sure it's working correctly. We'll use a tool called Postman to send requests to your API and examine the responses.

1. Install and Set up Postman

- Go to the official Postman website (www.postman.com/downloads/) and download the Windows version.
- Once the download is complete, run the installer and follow the on-screen instructions to install Postman on your computer. You can accept the defaults during installation. If you are asked about creating a workspace you can select "My Workspace".
- After installation, open the Postman application. You might be asked to create an account or sign in. You can skip this for now or create a free account if you'd like.

2. Getting Started with Postman

- Postman is an essential tool for testing APIs. Here's how to navigate its interface to test your library API effectively:
 - Request Builder: At the top of the workspace, this is where you'll configure your API request by selecting the method (e.g., GET or POST), entering the API URL, and adding headers or a request body as needed.
 - Send Button: Click this to send the configured request to your API.
 - Response Viewer: Below the request builder, you'll see the API's response. This includes the status code (e.g., 200 OK) and the response body in an easy-to-read format.
- Take a moment to explore these areas—they'll be your primary tools for interacting with your API and verifying its functionality

3. Testing GET Requests

Test the endpoint that retrieves all books from your API (`GET /books/`). This is like sending a request to your digital librarian to see the entire book collection.

- In Postman, click the "+" button to open a new request tab.
- Select the "GET" method from the dropdown.
- In the address bar, enter the URL for your API endpoint (e.g., `http://127.0.0.1:5000/books/`). Make sure your Flask application is running before sending the request.
- Click the "Send" button.
- You should see a response in the lower pane with a list of all books in JSON format. Verify that the status code is 200 OK, which indicates a successful request. If you don't get a 200 OK response, double-check that your Flask application is running, the URL is correct, and there are no errors in your API code. The response body should contain a list of all books in your library, formatted in JSON. Postman will usually format the JSON to make it easier to read.

Testing POST Requests

Test the endpoint that allows you to add a new book to your library (`POST /books/`). This is like sending a request to your digital librarian to add a new book to their collection.

- In the same Postman request tab, change the method from "GET" to "POST" using the dropdown menu.

- In the "Body" tab, select "raw" and choose "JSON" as the format.
- Enter the details of the new book in JSON format, like this:

1

2

3

4

5

- Click "Send." If everything is set up correctly, you should get a success message.

5. Interpreting Responses

- Postman shows you the status code of the response (i.e., 200 for success, 400 for a bad request).
- It also displays the response data in a formatted way, making it easy to understand.

6. Setting up Authentication (Basic Auth)

Some of your API endpoints, like adding a new book, might require authentication to restrict access to authorized users. This is like ensuring that only librarians can add new books to the library. Postman provides a convenient way to include authentication credentials in your requests. Here's how to set up Basic Auth in Postman:

- In your Postman request tab, click on the "Authorization" tab.
- Select "Basic Auth" from the "Type" dropdown.
- Enter the username and password you've set up for your API.
- Now, when you send the request, Postman will include the authentication credentials. If you get a 401 Unauthorized error even after setting up authentication, double-check that you've entered the correct username and password, and that your API's authentication logic is working as expected.

To view the solution files for this exercise, download and extract this archive:

1. You're sending a GET request to '/books/' endpoint. What would you expect in the response? Select the best answer.

1 / 1 point

A JSON object containing information about all the books in the library.

A 404 error because the endpoint is not defined.

A message indicating that you need authentication.

A single book object based on the provided ID.



Correct

Correct! This endpoint is designed to list all books.

2. You need to add a new book to the library's API. Which HTTP method and endpoint should you use? Select the best answer.

1 / 1 point

PUT '/books/'

POST '/books/'

DELETE '/books/'

GET '/books/'



Correct

Correct! The POST method is used to create new resources, and '/books/' is the correct endpoint.

3. You're sending a GET request to '/books/?search=Douglas'. What will the API return? Select the best answer.

1 / 1 point

A 400 error because the search parameter is not recognized.

All books in the library.

A message indicating that you need authentication.

A JSON object containing information about books written by Douglas Adams.



Correct

Correct! The search query is designed to filter books by title or author.

4. You're trying to add a new book but get a 401 error. What's the likely cause? Select the best answer.

1 / 1 point

You didn't provide the required book information.

You used the wrong HTTP method.

The server is down.

You didn't provide the correct authentication credentials.



Correct

Correct! The API requires authentication (username and password) to add new books.

5. You send a GET request to '/books/5' but the book with ID 5 doesn't exist. What response do you expect? Select the best answer.

1 / 1 point

A 500 Internal Server Error.

A 401 Unauthorized error.

A 200 OK response with an empty JSON object.

A 404 error with a message indicating that the book was not found.



Correct

Correct! The API returns a 404 error when a requested book ID doesn't exist.

6. Why is Postman a valuable tool for working with APIs? Select the best answer.


1 / 1 point

It allows you to easily send requests to APIs and view the responses.

It's the only way to test if an API is working.

It automatically writes API code for you.

It's only useful for developers who are new to APIs.

 **Correct**
Correct! Postman simplifies the process of interacting with APIs.