Authentication

Activity-1:

Description : Basic Authentication (https://swagger.io/docs/specification/authentication/basic-authentication/) for the Books API (https://github.com/mysilver/COMP9321-Data-Services/blob/master/Week6_Flask2/activity_3.py)

Steps :

1. Make a copy of the books API (https://github.com/mysilver/COMP9321-Data-Services/blob/master/Week6_Flask2/activity_3.py)
2. Add a python decoder for Basic Authentication (https://aviaryan.in/blog/gsoc/auth-flask-done-right.html) ; the authentication decoder must only authorize a user with the following credentials:

```
username='admin', password='admin'
```

3. Add the decorator on top of all of the resources' methods (get, post, put, delete methods)
4. Create a python script file; and do the following to see if the authentication is working with valid username and password:
   - send a get request to get (http://docs.python-requests.org/en/master/user/authentication/) the book with ID=206
5. Create a python script file; and do the following to see if the authentication is working with valid username and password:
   - send a get request to get (http://docs.python-requests.org/en/master/user/authentication/) the book with ID=206

```
Note: Unfortunately, the basic authentication is not supported by the automatic sw
generated by Flask-RestPlus
```

(https://github.com/mysilver/COMP9321-Data-Services/blob/master/Week8_Authentication/activity_1.py) (https://github.com/mysilver/COMP9321-Data-Services/blob/master/Week8_Authentication/activity_1_client.py)

Activity-2:

Description : In this activity you are going implement two methods to encode/decode credential information (username, password) to/from a json/string. This will later be used in the 3rd activity.

Steps:

1. Pick a quite long string (a private key) for encoding and decoding purposes.

2. Create a method which accepts a parameter called "username" : the method must create a json like {'username': 'admin', 'creation_time':XXXXX} then use the itsdangerous's json-web-signatures (https://pythonhosted.org/itsdangerous/#json-web-signatures) to encode it into a string with chosen private key
3. Create another method which accepts a token generated by the previous step
    - it returns the username by decoding the token if it a valid token and if token is created no later than 10 seconds.
    - otherwise, it returns proper exceptions
4. Create an encrypted token for the valid username and password
5. Decode the token to print the username("admin")
6. Wait for 10 seconds and try to decode the token again
7. Create a random string and try to decode it

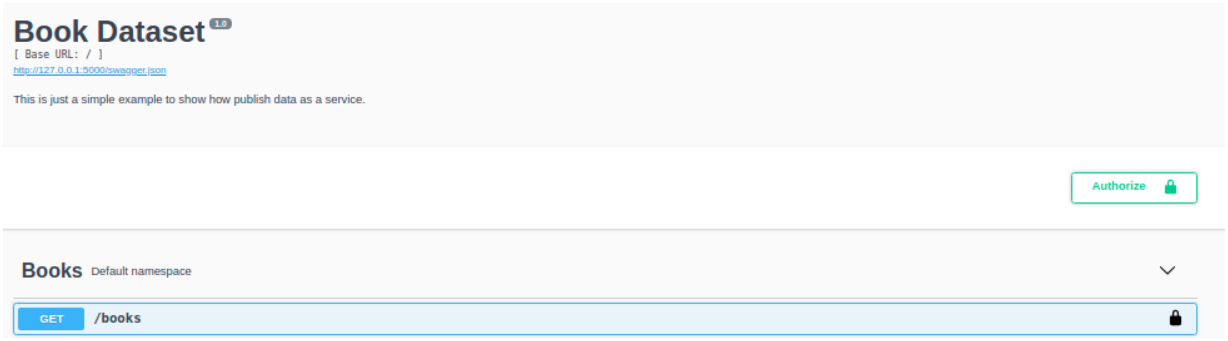
</> (https://github.com/mysilver/COMP9321-Data-Services/blob/master/Week8_Authentication/activity_2.py)

Activity-3:

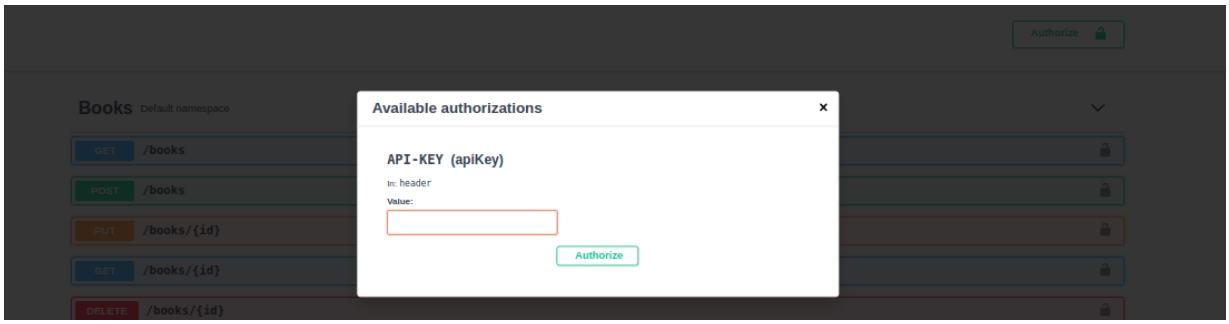
Description : Token-based Authentication (https://auth0.com/learn/token-based-authentication-made-easy/) for the Books API (https://github.com/mysilver/COMP9321-Data-Services/blob/master/Week6_Flask2/activity_3.py) . You can learn more about Token-Based Authentication by reading the following documentation : https://webcms3.cse.unsw.edu.au/COMP9321/18s1/resources/16047 (https://webcms3.cse.unsw.edu.au/COMP9321/18s1/resources/16047) .

Steps :

1. Create a new resource and implement an endpoint for authenticating and generating an authentication token based on the code written in the previous activity.
2. Add a python decoder (https://aviaryan.in/blog/gsoc/auth-flask-done-right.html) as you did in the first activity, but this time decode the given token (assumed to be present in request header) to see it is a valid token (decodable, and not expired);
3. Add the decorator on top of all of the resources' methods (get, post, put, delete methods)
4. Now you need to add authentication to the auto-generated swagger-doc (https://flask-restplus.readthedocs.io/en/stable/swagger.html#documenting-authorizations)
5. Test the authentication method, by running the RESTful service:
    1. Generate a token by invoking the endpoint generated in the first step
    2. Feed the generated token to the endpoints by clicking on the "Authorize" button and filling the input:



Then:



Now you can use the endpoints

(https://github.com/mysilver/COMP9321-Data-Services/blob/master/Week8_Authentication/activity_3.py)

Resource created about a month ago (Monday 14 March 2022, 03:07:09 PM), last modified 21 days ago (Tuesday 05 April 2022, 03:31:09 PM).

Comments

(/COMP9321/22T1/forums/search?forum_choice=resource/74093)

(/COMP9321/22T1/forums/resource/74093)

Add a comment

There are no comments yet.