Todo App using Flask in Python

Source: https://stackabuse.com/building-a-todo-app-with-flask-in-python/

Objective – create an API or web service for the given todo app using the REST-based architecture.

There are 4 HTTP verbs which each request to the RESTful system commonly uses. They are:

GET: Get a specific resource or a collection of resources

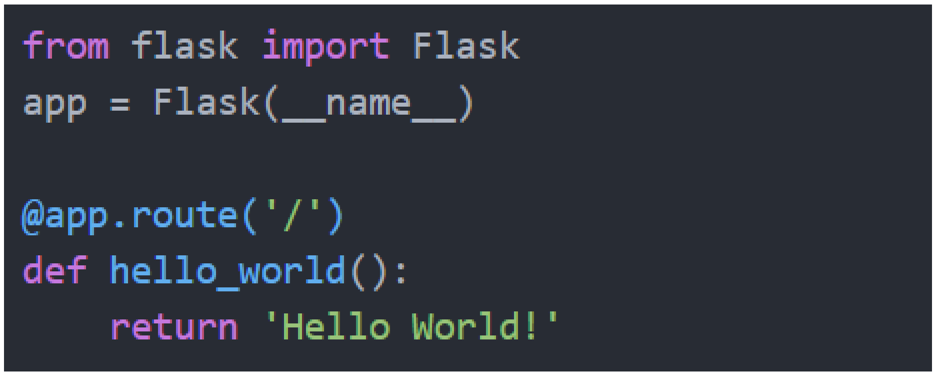
POST: Create a new resource

PUT: Update a specific resource

DELETE: Remove a specific resource

STEPS:

1. Install Flask using pip: $ pip install Flask
2. Then we created a file main.py and typed this command:



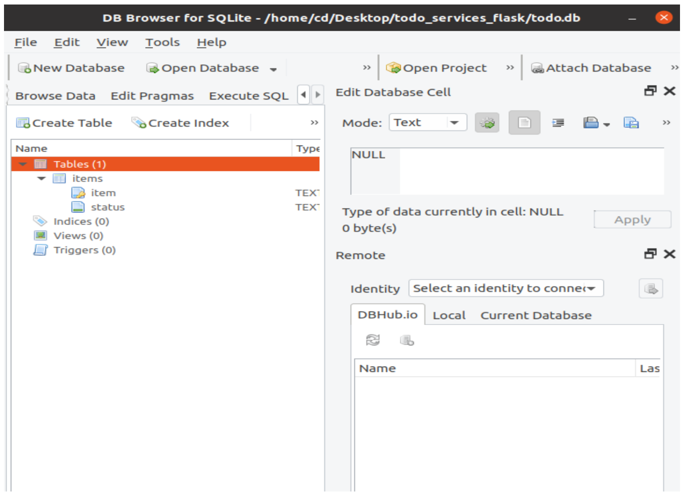
```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'
```

3. Once Flask is imported, we set up a route next. A URL pattern, HTTP method and function is used by a route which receives and handles the HTTP request.
4. Using the command $ FLASK_APP=main.py flask run we're able to run the Flask app.

NOTE: Once the Flask runs open a new terminal to use the cURL. This is important because if you try using curl on the same terminal the Flask will end.

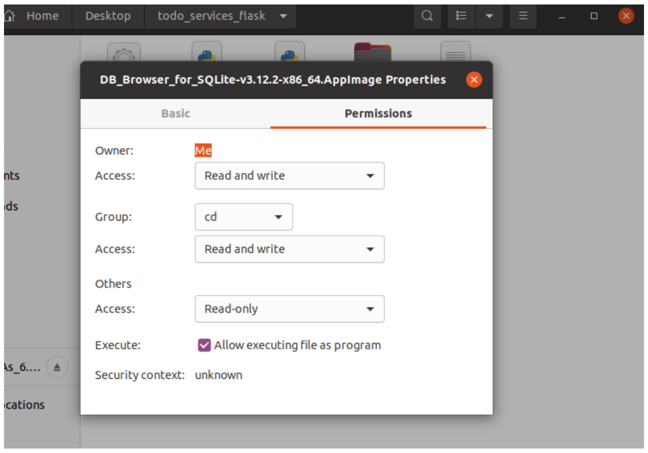
5. Use the command $ curl -X GET http://127.0.0.1:5000/ to run the cURL. The result will read "Hello World".

6. Then install the DB Browerser for SQLite to create a database. Once the database is created name it todo.db. The database should look like the figure below:



Type of data currently in cell: NULL
0 byte(s)

NOTE: If unable to load the database on the virtual machine, change the permissions as shown:



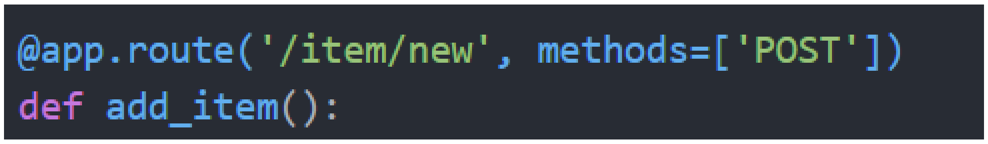
Owner: Me
Access: Read and write
Group: cd
Access: Read and write
Others
Access: Read-only
Execute: ✓ Allow executing file as program
Security context: unknown

7. Then using the given code, create the helper functions in a different file. Name the file helper.py.
8. Once done also update the main.py with the given code.
9. Then use the cURL command `$ curl -X POST http://127.0.0.1:5000/item -d'{"item": "Setting up Flask"}' -H 'Content-Type: application/json'` (incorrect link in online documentation).

NOTE:

i. Make sure after each change to the files you save it. Then go to the first terminal and stop the Flask app using ctrl C. Once done, restart the Flask to apply the new changes

ii. The route is calling item/new but the link given just has item and needs to be item/new

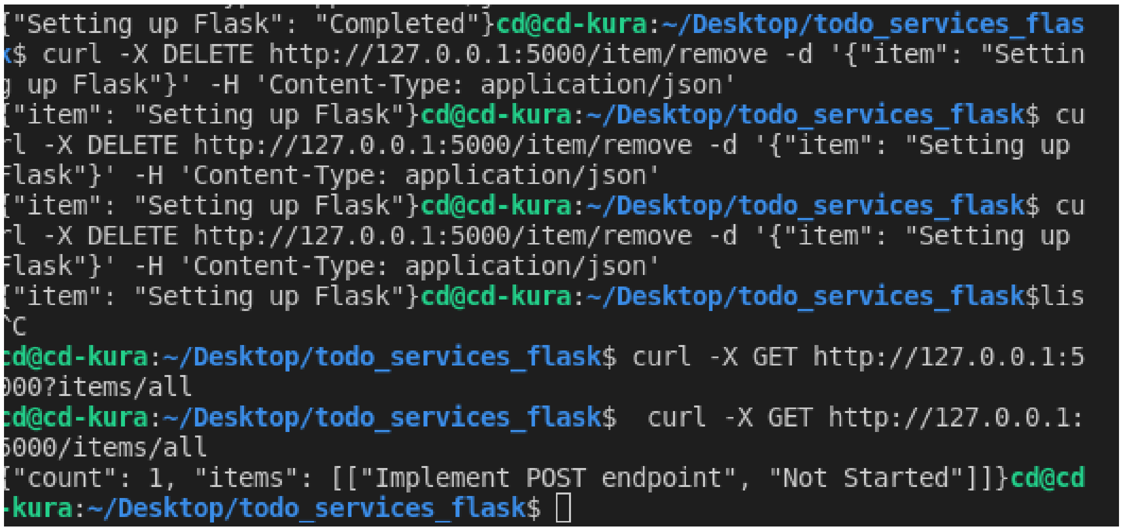


```
@app.route('/item/new', methods=['POST'])
def add_item():
```

Therefore, the correct curl link is $ curl -X POST http://127.0.0.1:5000/item/new -d'{"item": "Setting up Flask"}' -H 'Content-Type: application/json'.

10. Once successful you'll get the response {"Setting up Flask": "Not Started"}.
11. After, use the command $ curl -X POST http://127.0.0.1:5000/item -d '{"item": "Implement POST endpoint"}' -H 'Content-Type: application/json'.
12. Once successful you'll get the response {"Implement POST endpoint": "Not Started"}.
13. Then we used cURL to fetch the items and test our route. Use the cURL command $ curl -X GET http://127.0.0.1:5000/items/all.
14. Once successful you'll get the response json {"count": 2, "items": [["Setting up Flask", "Not Started"], [Implement POST endpoint", "Not Started"]]}.
15. Then use the cURL command $ curl -X GET http://127.0.0.1:5000/item/status?name=Setting+up+Flask.
16. Once successful you'll get the response {"status": "Not Started"}.
17. Then use another cURL command to test the route. Use the command $ curl -X PUT http://127.0.0.1:5000/item/update -d '{"item": "Setting up Flask", "status": "Completed"}' -H 'Content-Type: application/json'.
18. Once successful you'll get the response {"Setting up Flask": "Completed"}.
19. Now update both files using the codes needed to delete items. Then use the cURL command $ curl -X DELETE http://127.0.0.1:5000/item/remove -d '{"item": "Setting up Flask"}' -H 'Content-Type: application/json'.
20. Once successful you'll get the response {"item": "Temporary item to be deleted"} as shown in the figure below:

NOTE: I used the cURL command $ curl -X GET http://127.0.0.1:5000/items/all. The output I got was {"count": 1, "items": [["Implement POST endpoint", "Not Started"]]}. This let me know it was successful.



```
{"Setting up Flask": "Completed"}cd@cd-kura:~/Desktop/todo_services_flas
k$ curl -X DELETE http://127.0.0.1:5000/item/remove -d '{"item": "Settin
g up Flask"}' -H 'Content-Type: application/json'
{"item": "Setting up Flask"}cd@cd-kura:~/Desktop/todo_services_flask$ cu
rl -X DELETE http://127.0.0.1:5000/item/remove -d '{"item": "Setting up
Flask"}' -H 'Content-Type: application/json'
{"item": "Setting up Flask"}cd@cd-kura:~/Desktop/todo_services_flask$ cu
rl -X DELETE http://127.0.0.1:5000/item/remove -d '{"item": "Setting up
Flask"}' -H 'Content-Type: application/json'
{"item": "Setting up Flask"}cd@cd-kura:~/Desktop/todo_services_flask$lis
^C
cd@cd-kura:~/Desktop/todo_services_flask$ curl -X GET http://127.0.0.1:5
000?items/all
cd@cd-kura:~/Desktop/todo_services_flask$  curl -X GET http://127.0.0.1:
5000/items/all
{"count": 1, "items": [["Implement POST endpoint", "Not Started"]]}cd@cd
-kura:~/Desktop/todo_services_flask$ 
```