Software Engineer Intern - Services - Postman

Bengaluru

Satish Kumar

supervised by
Mr. Vivek Thuravupala

December 8, 2018

Introduction

In this project, I created an application similar to Twitter. In this application, I created different features. I create Unit/Integration tests for all functionality. I created this aplication in Python and Flask. This application have two files main.py and test.py. In main.py, I have all the functionality. In test.py, I have all the test cases.

Features

- login
- signup
- follow
- unfollow
- create tweet
- read tweet
- delete tweet
- like tweet
- unlike tweet
- retweet

Database structure

I implemented 3 table.

- user
- tweet
- reply

user

In user table, It will created when user will signup.

The fields are

```
"userName": userName, "userPassword": userPassword,
 "follow":[], "tweet":[], "like":[], "unlike":[],
 'messages':[]
```

tweet

In tweet table, It will created when user tweet.

The fields are

```
"title":title ,"message": tweet, "date":datetime.now(),
 "userId":result['_id'], 'retweet':[], 'likes': 0
```

reply

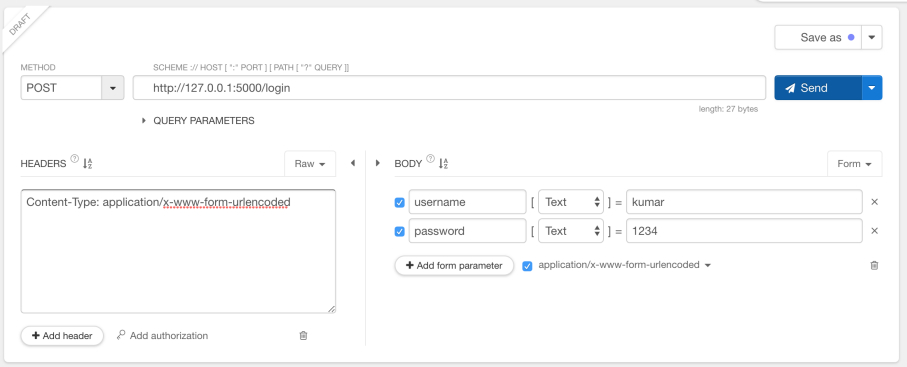
In reply table, It will created when user reply.

The fields are

```
"message": tweet, "date":datetime.now(),
'messageFrom':userid, 'messageTo':userid
```

Design

I used Rest client api for http request. I used post and get methods. And header type is **Content-Type: application/x-www-form-urlencoded** .
I used body as form data type. Visualization is follow for login:



We can see, I used two parameter username and password. Their types are text and values are kumar and 1234. I used body as form data. And url is **http://127.0.0.1:5000/login**. Flask run on **http://127.0.0.1:5000/**.
We have header and body in form have same for all funtions.

For each functionality, We have to need 3 differnt things.

- Method
- url
- parameters

Basic functionality

login

In login, User can login only if user is registered. And only one user can login in one device. And session for user login is 5 minute. If user is registered successfully and user login successfully, They will receive message Hello, ¡username¿. For login, we need

- method = post
- url = http://127.0.0.1:5000/login
- parameters = username and password

signup

In signup, User can signup only if user have unique name and any user not login. If user is registered successfully. For registeration, User need only (unique username) and password. For signup, we need

- method = post
- url = http://127.0.0.1:5000/signup
- parameters = username and password

logout

In logout, User have to be logged in. I created this functionality for checking the session. For logout, we need

- method = get
- url = http://127.0.0.1:5000/logout

Extended Functionality

follow

In follow, User can follow another user only if user is login and another user is exist. For follow, User need only username (which is the name of another user which user want to follow). For follow, we need

- method = post
- url = http://127.0.0.1:5000/follow
- parameters = username

unfollow

In unfollow, User can unfollow another user only if user is login, another user is exist and user is following it. For unfollow, User need only username (which is the name of another user which user want to unfollow). For unfollow, we need

- method = post
- url = http://127.0.0.1:5000/unfollow
- parameters = username

create tweet

In create tweet, User can create tweet only if user is login, and tweet title is unique. For create tweet, User need title and message. For create tweet, we need

- method = post
- url = http://127.0.0.1:5000/create
- parameters = title and message

read tweet

In read tweet, User can read tweet only if user is login, and tweet is exist. For read tweet, User need title (which is tweet title). For read tweet, we need

- method = post
- url = http://127.0.0.1:5000/read
- parameters = title

delete tweet

In delete tweet, User can delete tweet only if user is login, and tweet is exist and same user created it. For delete tweet, User need title (which is tweet title). For delete tweet, we need

- method = post
- url = http://127.0.0.1:5000/delete
- parameters = title

Unit/Integration tests

I created another file for testing is called test.py. But for testing firstly you have to run flask from main.py. And after you can run tests.

Steps

- open terminal
- cd app
- python test.py

Extra Credit

like tweet

In like tweet, User can like tweet only if user is login, tweet is exist and user is not liked it already. For like tweet, User need title (which is tweet title). For like tweet, we need

- method = post
- url = http://127.0.0.1:5000/like
- parameters = title

unlike tweet

In unlike tweet, User can unlike tweet only if user is login, tweet is exist and user is not unliked it already. For unlike tweet, User need title (which is tweet title). For unlike tweet, we need

- method = post
- url = http://127.0.0.1:5000/unlike
- parameters = title

retweet

In retweet, User can retweet only if user is login, tweet is exist. For retweet, User need title (which is tweet title). I did not implement retweet to retweet. For retweet, we need

- method = post
- url = http://127.0.0.1:5000/retweet
- parameters = title and message

reply

In reply, User can reply only if user is login, another user is exist. For reply, User need username (which is message to user) and message. For reply, we need

- method = post
- url = http://127.0.0.1:5000/reply
- parameters = username and message

Implementation

Requirement

- Python 3
- Flask link
- pymongo link(Mongodb for python)

Start the application

If all the requirement is installed and work properly. After that follow this steps

- open terminal

- cd app
- python main.py

And now you can use REST api.

Created by
Satish kumar