Twitter Followers Documentation

Release 0.0.1

Michal Dyzma

May 28, 2019

Contents

Last modified:	May 28, 2019

Application displaying second row followers on Twitter. Application uses Twitter REST API to connect with the page and download information necessary to build table.

CHAPTER 1

Page contents:

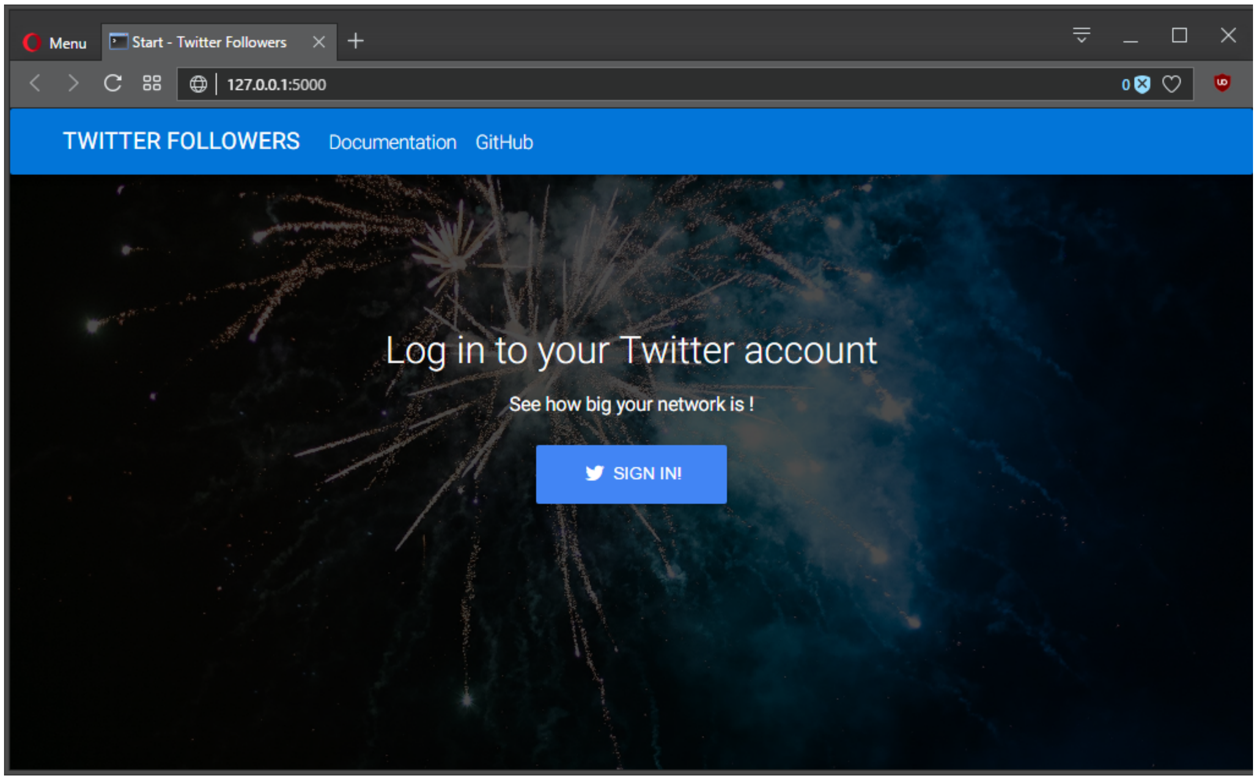
1.1 Quickstart

A Python app, which display followers of the followers of your Twitter account. Can be deployed to Heroku.

Application havily inspired by Miguel’s Grinberg Flask oAuth tutorial. UI based on MDB Free and EBM Bootstrap Plugin, available under MIT License and provided by MDBoostrap.com.

1.1.1 Usage

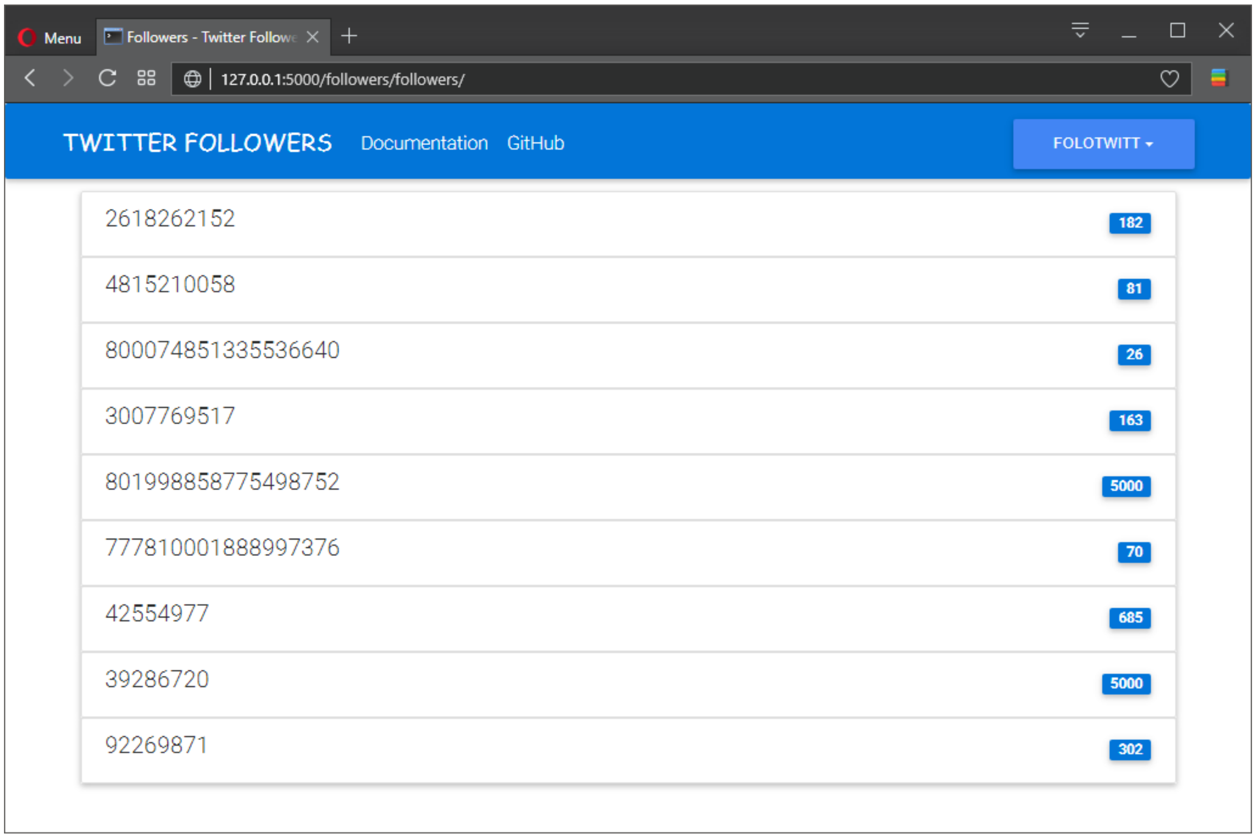
1. Visit [web site]()



Log in to your Twitter account

See how big your network is !

2. Sign in with your Twitter account
3. Browse list of followers who follow followers :)


Menu
Followers - Twitter Followe
127.0.0.1:5000/followers/followers/
TWITTER FOLLOWERS
Documentation
GitHub
FOLOTWITT

2618262152	182
4815210058	81
800074851335536640	26
3007769517	163
801998858775498752	5000
777810001888997376	70
42554977	685
39286720	5000
92269871	302

1.1.2 Deploying to Heroku

To deploy your copy of application simply click

Deploy to Heroku

> **Warning:** You will need Twitter app and SECRET CREDENTIALS provided by Twitter in order to make it work. See _twitter_dev

You can also quickly deploy using git. Make sure you have Python and HerokuCLI installed.:

```
$ git clone https://github.com/mdyzma/twitter_follo.git
$ cd twitter_follo
$ heroku create
$ git push heroku master
$ heroku open
```

1.1.3 Running Locally

It can be also run locally. Make sure you have Python installed properly. Also, install the HerokuCLI.

```
$ git clone https://github.com/mdyzma/twitter_follo.git
$ cd twitter_follo
$ pip install --no-cache-dir -r requirements.txt
$ heroku local
```

Your app should now be running on http://127.0.0.1:5000.

1.1.4 Documentation

For more information, i.e. about app further development used approaches, see project's documentation.

1.2 Application description

1.2.1 SDD Specification

There are two stories to implement: As a user I want to see followers of my followers on a page. Steps:

1. I go to "/" page
2. I see "Connect Twitter account" button
3. I press "Connect Twitter account" button
4. I see Twitter popup, I follow oAuth flow authorizing app
5. I get redirected to "/followers/followers" page
6. **I see a list of my "2nd line" followers. Each contains**
    - A Twitter handle, like "@madonna"

  - A number of my followers, that this person follows (example, Bob and Rick follow me, Andrew follows both of them, the number is 2).
  - The list doesn't contain my direct (1st line) followers

As a technical user I want to see followers of my followers as a JSON response Starting from where last scenario ended 1. I see a URL to JSON endpoint, that displays the list. 2. I go to the URL (no auth or session required) 3. I see a JSON response, which contains a list of my "2nd line" followers. Each contains:

- A Twitter handle, like "@madonna"
- A number of my followers, that this person follows

App should be: Heroku-deployable (please provide a link to the working deployment in the README) At least slightly documented Version controlled

:: 1. GET verify_cred (me) 2. count_#_followers 3. if lower than 5000: GET list followers ids named list_of_ids

4. Estimate number of gets/time (round up)
4. **while number of gets:**

   4.1. if number of gets less than 15: 4.1.1. GET list of followers ids from coursor page 4.1.2. update coursor 4.1.3. return to point 4.1.1

   4.2. if number of gets more than 15 (long data aquisition): 4.2.1. get list of followers ids from coursor page 4.2.2. update coursor 4.2.3. sleep 1 min 4.2.4. return to point 4.2.1
5. for item in

{4: {1: [89,4], 2: [454]}, 6768787: {1: [, 6, 7]}

There are two stories to implement: As a user I want to see followers of my followers on a page. Steps: 1. I go to "/" page 2. I see "Connect Twitter account" button 3. I press "Connect Twitter account" button 4. I see Twitter popup, I follow oAuth flow authorizing app 5. I get redirected to "/followers/followers" page 6. I see a list of my "2nd line" followers. Each contains

- A Twitter handle, like "@madonna"
- A number of my followers, that this person follows (example, Bob and Rick follow me, Andrew follows both of them, the number is 2).

The list doesn't contain my direct (1st line) followers

As a technical user I want to see followers of my followers as a JSON response Starting from where last scenario ended

1. I see a URL to JSON endpoint, that displays the list.
2. I go to the URL (no auth or session required)
3. **I see a JSON response, which contains a list of my "2nd line" followers. Each contains:**
   - A Twitter handle, like "@madonna"
   - A number of my followers, that this person follows

App should be: Heroku-deployable (please provide a link to the working deployment in the README) At least slightly documented Version controlled

One thing you might find troublesome is Twitter limiting API requests rate. This is part of the task, to decide what to do in such situation. Feel free to think through different solutions, but pick and implement only one (we will discuss them though).

1.2.2 Database

Data are stored in sqlite file.

def _print_report(user_obj, followers_list): print("Name: {}".format(user_obj.name)) print("Screen Name: {}".format("".join(["@", user_obj.screen_name]))) print("Created at: {}".format(user_obj.created_at.isoformat())) print() print("Followers:") print("-"*60) print("{:20}{:20}{:>20}".format("id", "@handle", "# of followers")) print("-"*60) for follower in followers_list:

print("{:20}{:20}{:20}".format(follower.id_str, "".join(["@", follower.screen_name]), follower.followers_count))

print("-"*60) print("nn")

1.3 Twitter Followers Workflow

This document contains some best practices, and basic workflow which can be used during further Twitter Followers app development.

1.3.1 Start

If you want to try this example, follow this steps:

1. Clone or download the project's repository: GitHub
2. Create a virtual environment and install packages from the requirements.txt file (you can use Python 2.7 or 3.5+).
3. Register "app" with Twitter on https://apps.twitter.com.
4. Update local_settings.py with the ids and secret codes of your Twitter app.
5. Run app locally

1.3.2 Clone or fork repository

Get source files from Git repository from the server

```
$ git clone https://gitlab.com/mdyzma/twitter_follo.git
```

1.3.3 Virtual environment and dependencies

It is a good practice to develop project in clean environment, to control all dependencies. This step requires Python and virtualenv package to be present in your system Python installation. Install `virtualenv` using python package manager.

```
$ pip install virtualenv -U
```

It is also good to keep your environments away from the project folder. it helps to avoid having large folders (i.e. python envs with installed dependencies, sometimes hundreds of MB) included in version control system. Usually I create one hidden folder in my users home directory i.e. `.envs/` and place all virtual environments for each project there.

```
$ mkdir ~/.envs
$ virtualenv ~/.envs/twitter
```

This will create python virtual environment with the specified name in `.envs` folder located in users home directory (Linux only). Check how to manage virtualenvs on windows OS here

To activate it

```
$ source ~/.envs/bin/activate

(twitter) $
```

Prompt should change and be preceded with the environment name in parenthesis. Once fresh environment is ready it is time to install development dependencies:

```
(twitter) $ pip install --no-cache-dir -r requirements/dev.txt
```

1.3.4 Continuous Integration

Project can be easly connected to CI service like TravisCI or CircleCI and authomaticaly run tests and deploy code to Heroku. Github repository can be also be directly connected to Heroku service. In that case All services will sense specified git branch changes and runjobs specified in configuration file.

Configuration file for TravisCI is located in project's root directory: `travis-ci.yml`

```
language: python
    python:
        - "2.7"
        - "3.5"
        - "3.6"
        - "nightly" # currently points to 3.7-dev
# command to install dependencies
install: "pip install -r requirements.txt"

# command to run tests and coverage report
script: python -m pytest --cov-report xml --cov=app tests/
script: codecov

deploy:
    provider: heroku
api_key:
    secure: "API-key-provided-by-heroku"

app: fast-forest-95874
```

Warning: Replace API key and name of the app with our own.

1.3.5 Manual Testing

Application is developed in Ext programming approach (extremely late working hours). Tests are grouped in root folder in `tests/`. Tests use `pytest` module.

To run entire suite use

```
(tweeter) $ python -m pytest --cov=src tests/
```

Output similar to this will appear

```
============================= test session starts ==============================
platform linux2 -- Python 2.7.13, pytest-3.1.0, py-1.4.33, pluggy-0.4.0
rootdir: /home/mdyzma/projects/twitter_follo, inifile:
plugins: cov-2.3.1
collected 1 items

tests/test_twitter.py E

==================================== ERRORS ====================================
_____________________ ERROR at setup of test_api_connection ____________________

    @pytest.fixture
    def api():
>       api = auth.main()
E       AttributeError: 'module' object has no attribute 'main'

tests/test_twitter.py:28: AttributeError
=========================== 1 error in 0.53 seconds ============================
ERROR: Failed to generate report: No data to report.
```

First run of test SHOULD fail. Use the force of TDD to make it nice and green.

For pretty html report add flag

```
(twitter) $ python -m pytest --cov-report html --cov=src`
```

The later will produce nicely formatted report in `htmlcov/` folder.

1.3.6 Code quality

I use PyCharm built-in pylint, but there is `pylint` module in python, which shows where code deviates from the official guidelines. See typical report:

```
(twitter) $ pylint --report=y app.py

pylint --reports=y app.py
No config file found, using default configuration
************* Module twitter_follo.app
C:  1, 0: Missing module docstring (missing-docstring)
E:  3, 0: Unable to import 'flask_sqlalchemy' (import-error)
E:  4, 0: Unable to import 'flask_login' (import-error)
C:  8, 0: Invalid constant name "app" (invalid-name)
C: 11, 0: Invalid constant name "consumer_key" (invalid-name)
C: 12, 0: Invalid constant name "consumer_secret" (invalid-name)
C: 13, 0: Invalid constant name "callback_url" (invalid-name)
C: 15, 0: Invalid constant name "db" (invalid-name)
C: 16, 0: Invalid constant name "lm" (invalid-name)
C: 19, 0: Invalid constant name "session" (invalid-name)
C: 20, 0: Invalid constant name "data" (invalid-name)
C: 22, 0: Missing class docstring (missing-docstring)
C: 24, 4: Invalid class attribute name "id" (invalid-name)
R: 22, 0: Too few public methods (0/2) (too-few-public-methods)
```

(continues on next page)

(continued from previous page)

```
W: 31,14: Redefining built-in 'id' (redefined-builtin)
C: 31, 0: Invalid argument name "id" (invalid-name)
C: 31, 0: Missing function docstring (missing-docstring)
C: 36, 0: Missing function docstring (missing-docstring)
C: 41, 0: Missing function docstring (missing-docstring)
C: 52, 0: Missing function docstring (missing-docstring)
C: 78, 0: Missing function docstring (missing-docstring)
W:  4, 0: Unused login_user imported from flask_login (unused-import)
W:  4, 0: Unused logout_user imported from flask_login (unused-import)
W:  4, 0: Unused current_user imported from flask_login (unused-import)


Report
======
52 statements analysed.

Statistics by type
------------------

+---------+-------+-----------+-----------+------------+---------+
|type     |number |old number |difference |%documented |%badname |
+=========+=======+===========+===========+============+=========+
|module   |1      |1          |=          |0.00        |0.00     |
+---------+-------+-----------+-----------+------------+---------+
|class    |1      |1          |=          |0.00        |0.00     |
+---------+-------+-----------+-----------+------------+---------+
|method   |0      |0          |=          |0           |0        |
+---------+-------+-----------+-----------+------------+---------+
|function |5      |5          |=          |0.00        |0.00     |
+---------+-------+-----------+-----------+------------+---------+



External dependencies
---------------------
::

    auth (twitter_follo.app)
    flask (twitter_follo.app)
    tweepy (twitter_follo.app)



Raw metrics
-----------

+----------+-------+------+---------+-----------+
|type      |number |%     |previous |difference |
+==========+=======+======+=========+===========+
|code      |58     |69.05 |NC       |NC         |
+----------+-------+------+---------+-----------+
|docstring |0      |0.00  |NC       |NC         |
+----------+-------+------+---------+-----------+
|comment   |3      |3.57  |NC       |NC         |
+----------+-------+------+---------+-----------+
|empty     |23     |27.38 |NC       |NC         |
+----------+-------+------+---------+-----------+
```

(continues on next page)

(continued from previous page)

```


Duplication
-----------

+-------------------------+------+---------+-----------+
|                         |now   |previous |difference |
+=========================+======+=========+===========+
|nb duplicated lines      |0     |0        |=          |
+-------------------------+------+---------+-----------+
|percent duplicated lines |0.000 |0.000    |=          |
+-------------------------+------+---------+-----------+



Messages by category
--------------------

+-----------+-------+---------+-----------+
|type       |number |previous |difference |
+===========+=======+=========+===========+
|convention |17     |17       |=          |
+-----------+-------+---------+-----------+
|refactor   |1      |1        |=          |
+-----------+-------+---------+-----------+
|warning    |4      |4        |=          |
+-----------+-------+---------+-----------+
|error      |2      |2        |=          |
+-----------+-------+---------+-----------+



Messages
--------

+-----------------------+------------+
|message id             |occurrences |
+=======================+============+
|invalid-name           |10          |
+-----------------------+------------+
|missing-docstring      |7           |
+-----------------------+------------+
|unused-import          |3           |
+-----------------------+------------+
|import-error           |2           |
+-----------------------+------------+
|too-few-public-methods |1           |
+-----------------------+------------+
|redefined-builtin      |1           |
+-----------------------+------------+




------------------------------------------------------------------
Your code has been rated at 3.85/10 (previous run: 3.85/10, +0.00)
```

1.3.7 Documentation

For documentation reStructuredText plain text markup syntax is used. It is an easy-to-read, what-you-see-is-what-you-get markup, with large capabilities to automate documentation creation, or include it into CI pipeline.

To regenerate documentation after some changes cd to docs/ folder and type:

```
$ make html
```

Static web page will be created from `*.rst` files located in `source/` directory. Location of main static site is: `docs/build/html/index.html`. It works much better if it is served (it uses some basic javascript). Therefore there are two options.

1. Run external software to serve www (apache, nginx)
2. use built in python server from html folder.

There is also third possibility, which may be much better in case of large documentation updates. Install `sphinx-autobuild`

```
$ pip install sphinx-autobuild
```

Enter docs folder and run local server, which will rerun sphinx build process when it detects changes in `.rst` files. To start server type:

```
(twitter) $ sphinx-autobuild docs\source docs\build\html

+--------- manually triggered build ---------------------------------------------
| Running Sphinx v1.6.2
| loading pickled environment... failed: unsupported pickle protocol: 4
| building [mo]: targets for 0 po files that are out of date
| building [html]: targets for 6 source files that are out of date
| updating environment: 6 added, 0 changed, 0 removed
| reading sources... [ 16%] api
| reading sources... [ 33%] contribution
| reading sources... [ 50%] development
| reading sources... [ 66%] index
| reading sources... [ 83%] license
| reading sources... [100%] quickstart
|
| looking for now-outdated files... none found
| pickling environment... done
| checking consistency... done
| preparing documents... done
| reading sources... [ 16%] api
| reading sources... [ 33%] contribution
| reading sources... [ 50%] development
| reading sources... [ 66%] index
| reading sources... [ 83%] license
| reading sources... [100%] quickstart
|
| generating indices... genindex
| writing additional pages... search
| copying static files... done
| copying extra files... done
| dumping search index in English (code: en) ... done
| dumping object inventory... done
| build succeeded.
+--------------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```

[I 170522 13:12:16 server:283] Serving on http://127.0.0.1:8000
[I 170522 13:12:16 handlers:60] Start watching changes
[I 170522 13:12:16 handlers:62] Start detecting changes
```

To extract doctrings from source code sphinx autodoc extension is used. Default docstring format is numpydoc. To parse it properly napoleon project is used.

To generate documentation based on source code doctstrings type:

```
(twitter) $ sphinx-apidoc -f -o docs/source .

Creating file docs/source/twitter_follo.rst.
Creating file docs/source/modules.rst.
```

1.4 Contribution

You can contribute in many ways, but before you do please check this article: Twitter App Development

1.4.1 Report Bugs

Report bugs at https://github.com/mdyzma/twitter_follo/issues

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- If you can, provide detailed steps to reproduce the bug.
- If you don't have steps to reproduce the bug, just note your observations in as much detail as you can. Questions to start a discussion about the issue are welcome.

1.4.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

1.4.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "please-help" is open to whoever wants to implement it.

Please do not combine multiple feature enhancements into a single pull request.

Note: this project is very conservative, so new features that aren't tagged with "please-help" might not get into core. We're trying to keep the code base small, extensible, and streamlined. Whenever possible, it's best to try and implement feature ideas as separate projects outside of the core codebase.

1.4.4 Write Documentation

Twitter Display could always use more documentation, whether as part of the official docs, in docstrings, or even on the web in blog posts, articles, and such.

1.4.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/mdyzma/twitter_follo/issues

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

1.5 API

1.6 LICENSE

CHAPTER 2

Indices and tables

- genindex
- modindex
- search