Usage

Load tag like this:

{% load twitter\_tag %}

Get user's (futurecolors in example) most recent tweets and store them in tweets variable:

{% get\_tweets for "futurecolors" as tweets %}

Now you have a list of tweets in your template context, which you can iterate over like this:

<ul>

{% for tweet in tweets %}

<li>{{ tweet.html|safe }}</li>

{% endfor %}

</ul>

Installation

This app works with python 2.7 and 3.3, Django 1.4-1.6.

Recommended way to install is pip:

pip install django-twitter-tag

Add twitter\_tag to INSTALLED\_APPS in settings.py:

INSTALLED\_APPS = (...

'twitter\_tag',

...

)

Configuration

Twitter API 1.1 requires authentication for every request you make, so you have to provide some credentials for oauth dance to work. First, create an application, second, request access token on newly created app page. The process of obtaining a token is explained in detail in docs.

Here is an example of how your config might look like:

# settings.py

# Make sure to replace with your own values, theses are just made up

# Your access token: Access token

TWITTER\_OAUTH\_TOKEN = '91570701-BQMM5Ix9AJUC5JtM5Ix9DtwNAiaaYIYGN2CyPgduPVZKSX'

# Your access token: Access token secret

TWITTER\_OAUTH\_SECRET = 'hi1UiXm8rF4essN3HlaqMz7GoUvy3e4DsVkBAVsg4M'

# OAuth settings: Consumer key

TWITTER\_CONSUMER\_KEY = '3edIOec4uu00IGFxvQcwJe'

# OAuth settings: Consumer secret

TWITTER\_CONSUMER\_SECRET = 'YBD6GyFpvumNbNA218RAphszFnkifxR8K9h8Rdtq1A'

For best performance you should set up django cache framework. Cache is used both internally to store last successful json response and externally (see Caching below).

Examples

You can specify number of tweets to show:

{% get\_tweets for "futurecolors" as tweets limit 10 %}

To filter out tweet replies (that start with @ char):

{% get\_tweets for "futurecolors" as tweets exclude "replies" %}

To ignore native retweets:

{% get\_tweets for "futurecolors" as tweets exclude "retweets" %}

Or everything from above together:

{% get\_tweets for "futurecolors" as tweets exclude "replies, retweets" limit 10 %}

Search tag (experimental)

You can search for tweets:

{% search\_tweets for "python 3" as tweets limit 5 %}

Search api arguments are supported via key=value pairs:

{% search\_tweets for "python 3" as tweets lang='eu' result\_type='popular' %}

Relevant API docs for search.

Caching

It's strongly advised to use template caching framework to reduce the amount of twitter API calls and avoid reaching rate limit (currently, 180 reqs in 15 minutes):

{% load twitter\_tag cache %}

{% cache 60 my\_tweets %}

{% get\_tweets for "futurecolors" as tweets exclude "retweets" %}

...

{% endcache %}

Extra

Tweet's properties

get\_tweets returns a list of tweets into context. Each tweets is a json dict, that has exactly the same attributes, as stated in API 1.1 docs, describing tweet json. Tweet's created timestamp is converted to python object and is available in templates:

{{ tweet.datetime|date:"D d M Y" }}

Tweet's html

Tweet also has extra html property, which contains tweet, formatted for html output with all needed links. Note, Twitter has guidelines for developers on how embeded tweets should look like.

Exception handling

Any Twitter API exceptions like 'Over capacity' are silenced and logged. Django cache is used internally to store last successful response in case twitter is down.

Going beyond

Since version 1.0 you can create your own template tags for specific twitter queries, not supported by this library. Simply inherit from twitter\_tag.templatetags.twitter\_tag.BaseTwitterTag and implement your own get\_json method (tag syntax is contolled by django-classy-tags).

Development

To install development version, use pip install django-twitter-tag==dev

Tests

Run:

DJANGO\_SETTINGS\_MODULE = twitter\_tag