bottr Documentation

Release 0.1.4

Steven Lang

May 13, 2018

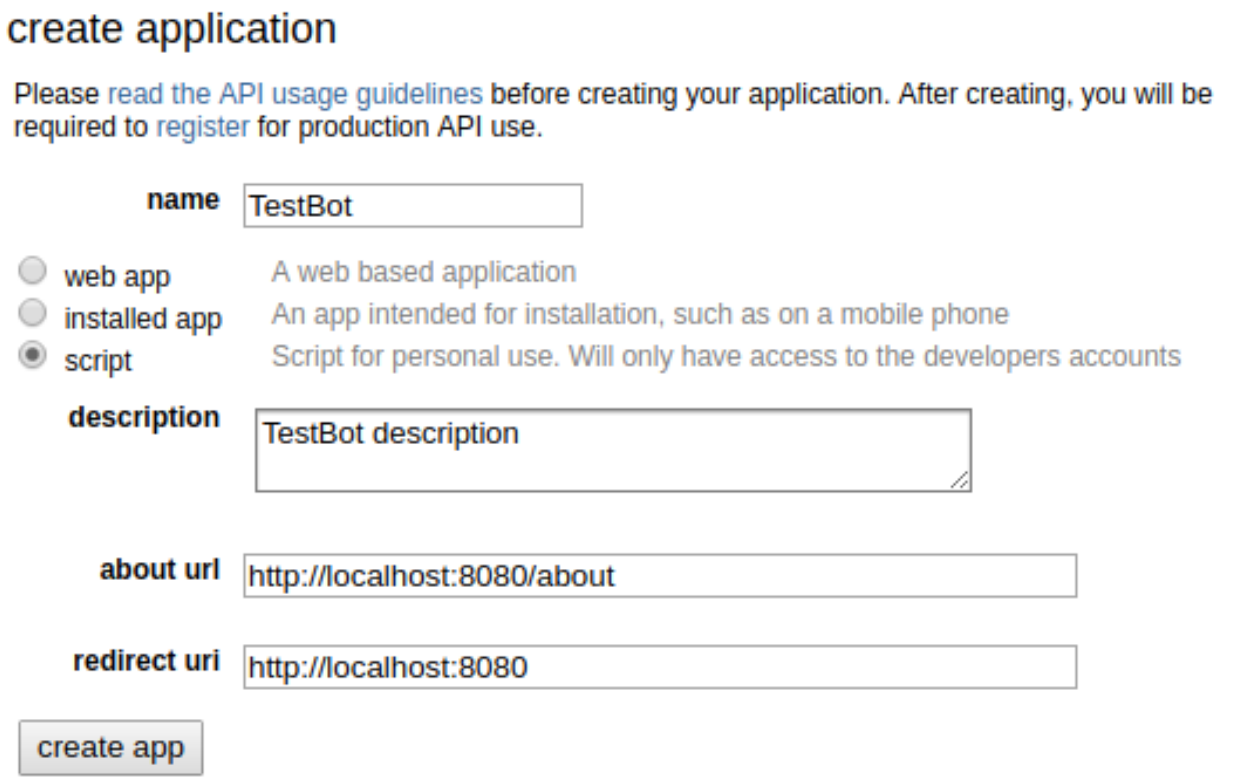
Contents:

Bottr is supposed to make writing bots for reddit easy. It relies on the *Python Reddit API Wrapper* PRAW.

CHAPTER 1

Bot Account Setup

To instantiate a `praw.Reddit` instance, you need to provide a few login credentials. If you have not done so already, go to https://www.reddit.com/prefs/apps and click the *create a new app* button. This should pop up the following form:



After filling out the inputs and pressing *create app*, you will see a new application in the list above:

With the information in this box it is now possible to create a `praw.Reddit` using the following parameters:

client_id The *personal use script* listed in the application box.

client_secret The *secret* listed in the application box.

username Username of the bot reddit account.

password Password of the bot reddit account.

user_agent User agent description, e.g. `Script by u/testbot`. See also the reddit api-rules.

For the above example, a reddit instance can be created as follows:

```
import praw
reddit = praw.Reddit(client_id='6TC26cMNLi-qaQ',
                     client_secret='vDY3bsgl8RWXMDil2HRjbD2EUBs',
                     password='botpassword',
                     user_agent='Script by u/test-bot',
                     username='test-bot')
```

Check out Authenticating via OAuth in the PRAW documentation for further details.

CHAPTER 2

Predefined Bots

2.1 Description

Bottr comes with a set of predefined bots, e.g. `CommentBot` or `SubmissionBot`. Both bots accepts a function as constructor argument. The bots listen to a stream of new comments/submissions and call the given function with a `praw.models.Comment` or `praw.models.Submission` object respectively.

The parsing function for comments or submissions might take some time, e.g. calling `praw.models.Comment.parent()` makes a new request to the reddit api and waits for a response. Therefore, it is possible to specify the argument `n_jobs` when creating the bots. This is the maximum number of comments/submissions that can be processed in parallel by the bot. The stream of new comments/submissions are internally put into a `Queue`, being available to a list of worker threads that successively poll new objects to process from the queue. The `n_jobs` argument defines how many worker threads are available.

2.2 Bots

2.2.1 Comment Bot

class bottr.bot.**CommentBot**(*reddit: praw.reddit.Reddit, name: str = 'CommentBot', func_comment: typing.Callable[[praw.models.reddit.comment.Comment], NoneType] = None, func_comment_args: typing.List = None, subreddits: typing.Iterable = None, n_jobs=4*)

This bot listens to incoming comments and calls the provided method `func_comment` as `func_comment(comment, *func_comment_args)` for each `comment` that is submitted in the given `subreddits`.

Creates a bot that listens to comments in a list of subreddits and calls a given function on each new comment.

Parameters

- **reddit** – `praw.Reddit` instance. Check *Bot Account Setup* on how to create it.
- **name** – Bot name

- **func_comment** – Comment function. It needs to accept a `praw.models.Comment` object and may take more arguments. For each comment created in `subreddits`, a `praw.models.Comment` object and all `fun_comments_args` are passed to `func_comment` as arguments.
- **func_comment_args** – Comment function arguments.
- **subreddits** – List of subreddit names. Example: `['AskReddit', 'Videos', ...]`
- **n_jobs** – Number of parallel threads that are started when calling *`start()`* to process in the incoming comments.

Example usage:

```
# Write a parsing method
def parse(comment):
   if 'banana' in comment.body:
       comment.reply('This comment is bananas.')

reddit = praw.Reddit(...) # Create a PRAW Reddit instance
bot = CommentBot(reddit=reddit, func_comment=parse)
bot.start()
```

start()
Starts this bot in a separate thread. Therefore, this call is non-blocking.

It will listen to all new comments created in the `subreddits` list.

stop()
Stops this bot.

Returns as soon as all running threads have finished processing.

2.2.2 Submission Bot

class bottr.bot.**SubmissionBot**(*reddit: praw.reddit.Reddit, name: str = 'SubmissionBot', func_submission: typing.Callable[[praw.models.reddit.comment.Comment], NoneType] = None, func_submission_args: typing.List = None, subreddits: typing.Iterable = None, n_jobs=4*)

Bottr Bot instance that can take a method `func_submission` and calls that method as `func_submission(submission, *func_submission_args)`

Can listen to new submissions made on a given list of subreddits.

Parameters

- **reddit** – `praw.Reddit` instance. Check here on how to create it.
- **name** – Bot name
- **func_submission** – Submission function. It needs to accept a `praw.models.Submission` object and may take more arguments. For each submission created in `subreddits`, a `praw.models.Submission` object and all `fun_submission_args` are passed to `func_submission` as arguments.
- **func_submission_args** – submission function arguments.
- **subreddits** – List of subreddit names. Example: `['AskReddit', 'Videos', ...]`

- **n_jobs** – Number of parallel threads that are started when calling *start()* to process in the incoming submissions.

Example usage:

```
# Write a parsing method
def parse(submission):
    if 'banana' in submission.title:
        submission.reply('This submission is bananas.')

reddit = praw.Reddit(...) # Create a PRAW Reddit instance
bot = SubmissionBot(reddit=reddit, func_submission=parse)
bot.start()
```

start()
Starts this bot in a separate thread. Therefore, this call is non-blocking.

It will listen to all new submissions created in the `subreddits` list.

stop()
Stops this bot.

Returns as soon as all running threads have finished processing.

2.2.3 Message Bot

class bottr.bot.**MessageBot**(*reddit: praw.reddit.Reddit, name: str = 'InboxBot', func_message: typing.Callable[[praw.models.reddit.message.Message], NoneType] = None, func_message_args: typing.List = None, n_jobs=1*)
This bot listens to incoming inbox messages and calls the provided method `func_message` as `func_message(message, *func_message_args)` for each `message` that is new in the inbox.

Parameters

- **reddit** – `praw.Reddit` instance. Check *Bot Account Setup* on how to create it.
- **name** – Bot name
- **func_message** – Message function. It needs to accept a `praw.models.Message` object and may take more arguments. For each new message in the inbox, a `praw.models.Message` object and all `fun_message_args` are passed to `func_message` as arguments.
- **func_message_args** – Message function arguments.
- **n_jobs** – Number of parallel threads that are started when calling *start()* to process in the incoming messages.

Example usage:

```
# Write a parsing method
def parse(message):
   message.reply('Hello you!')

reddit = praw.Reddit(...) # Create a PRAW Reddit instance
bot = MessageBot(reddit=reddit, func_message=parse)
bot.start()
```

start()
Starts this bot in a separate thread. Therefore, this call is non-blocking.

It will listen to all new inbox messages created.

stop()
Stops this bot.

Returns as soon as all running threads have finished processing.

CHAPTER 3

Bot Utilities

The module *bottr.util* provides some functions that can be helpful when handling certain situations while parsing comments or submission.

bottr.util.**handle_rate_limit**(*func: typing.Callable[[typing.Any], typing.Any], *args, **kwargs*) → typing.Any

Calls `func` with given arguments and handle rate limit exceptions.

Parameters

- **func** – Function to call
- **args** – Argument list for `func`
- **kwargs** – Dict arguments for *func*

Returns `func` result

bottr.util.**check_comment_depth**(*comment: praw.models.reddit.comment.Comment*, *max_depth=3*) → bool

Check if comment is in a allowed depth range

Parameters

- **comment** – `praw.models.Comment` to count the depth of
- **max_depth** – Maximum allowed depth

Returns True if comment is in depth range between 0 and max_depth

bottr.util.**get_subs**(*subs_file='subreddits.txt'*, *blacklist_file='blacklist.txt'*) → typing.List[str]

Get subs based on a file of subreddits and a file of blacklisted subreddits.

Parameters

- **subs_file** – List of subreddits. Each sub in a new line.
- **blacklist_file** – List of blacklisted subreddits. Each sub in a new line.

Returns List of subreddits filtered with the blacklisted subs.

Example files:

```
sub0
sub1
sub2
...
```

bottr.util.**init_reddit**(*creds_path='creds.props'*) → praw.reddit.Reddit
Initialize the reddit session by reading the credentials from the file at creds_path.

Parameters **creds_path** – Properties file with the credentials.

Example file:

```
client_id=CLIENT_ID
client_secret=CLIENT_SECRET
password=PASSWORD
user_agent=USER_AGENT
username=USERNAME
```

Check out bottr-template for a convenient code template to start with.

CHAPTER 4

Quick Start

The following is a quick example on how to monitor *r/AskReddit* for new comments. If a comment contains the string `'banana'`, the bot prints the comment information:

```
import praw
import time

from bottr.bot import CommentBot

def parse_comment(comment):
    """Define what to do with a comment"""
    if 'banana' in comment.body:
        print('ID: {}'.format(comment.id))
        print('Author: {}'.format(comment.author))
        print('Body: {}'.format(comment.body))

if __name__ == '__main__':

    # Get reddit instance with login details
    reddit = praw.Reddit(client_id='id',
                         client_secret='secret',
                         password='botpassword',
                         user_agent='Script by /u/...',
                         username='botname')

    # Create Bot with methods to parse comments
    bot = CommentBot(reddit=reddit,
                     func_comment=parse_comment,
                     subreddits=['AskReddit'])

    # Start Bot
    bot.start()

    # Run bot for 10 minutes
    time.sleep(10*60)
```

```
    # Stop Bot
    bot.stop()
```

Check out *Bot Account Setup* to see how to get the arguments for `praw.Reddit`.

Note: Please read the reddit bottiquette if you intend to run a bot that interacts with reddit, such as writing submissions/comments etc.

Python Module Index

b

Index