django-telegrambot Documentation

Release 1.0.1

django-telegrambot

Jun 01, 2017

Contents

Contents:

CHAPTER 1

django-telegrambot

A simple app to develop Telegram bots with Django

Documentation

The full documentation is at https://django-telegrambot.readthedocs.org.

Changelog

- **IMPORTANT ver 1.0.0** : If you upgrade from a previous version, you **MUST** change how to include `django_telegrambot.urls` and modify your `settings.py`.

Quickstart

Install django-telegrambot:

```
pip install django-telegrambot
```

Configure your installation

Add `django_telegrambot` in `INSTALLED_APPS`

```
#settings.py
INSTALLED_APPS = (
    ...
    'django_telegrambot',
```

```
    ...
)
```

And set your bots:

```
#settings.py
#Django Telegram Bot settings

DJANGO_TELEGRAMBOT = {

    'MODE' : 'WEBHOOK', #(Optional [str]) # The default value is WEBHOOK,
                        # otherwise you may use 'POLLING'
                        # NB: if use polling you must provide to run
                        # a management command that starts a worker

    'WEBHOOK_SITE' : 'https://mywebsite.com',
    'WEBHOOK_PREFIX' : '/prefix', # (Optional[str]) # If this value is specified,
                                  # a prefix is added to webhook url

    #'WEBHOOK_CERTIFICATE' : 'cert.pem', # If your site use self-signed
                         #certificate, must be set with location of your public key
                         #certificate.(More info at https://core.telegram.org/bots/
↪self-signed )

    'BOTS' : [
        {
           'TOKEN': '123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11', #Your bot token.

           #'ALLOWED_UPDATES':(Optional[list[str]]), # List the types of
                                                     #updates you want your bot to␣
↪receive. For example, specify
                                                     #``["message", "edited_channel_post
↪", "callback_query"]`` to
                                                     #only receive updates of these␣
↪types. See ``telegram.Update``
                                                     #for a complete list of available␣
↪update types.
                                                     #Specify an empty list to receive␣
↪all updates regardless of type
                                                     #(default). If not specified, the␣
↪previous setting will be used.
                                                     #Please note that this parameter␣
↪doesn't affect updates created
                                                     #before the call to the setWebhook,
↪ so unwanted updates may be
                                                     #received for a short period of␣
↪time.

           #'TIMEOUT':(Optional[int|float]), # If this value is specified,
                                     #use it as the read timeout from the server

           #'WEBHOOK_MAX_CONNECTIONS':(Optional[int]), # Maximum allowed number of
                                     #simultaneous HTTPS connections to the webhook for␣
↪update
                                     #delivery, 1-100. Defaults to 40. Use lower values␣
↪to limit the
                                     #load on your bot's server, and higher values to␣
↪increase your
```

```
                                    #bot's throughput.

            #'POLL_INTERVAL' : (Optional[float]), # Time to wait between polling␣
↪updates from Telegram in
                            #seconds. Default is 0.0

            #'POLL_CLEAN':(Optional[bool]), # Whether to clean any pending updates on␣
↪Telegram servers before
                                    #actually starting to poll. Default is False.

            #'POLL_BOOTSTRAP_RETRIES':(Optional[int]), # Whether the bootstrapping␣
↪phase of the `Updater`
                                    #will retry on failures on the Telegram server.
                                    #|   < 0 - retry indefinitely
                                    #|     0 - no retries (default)
                                    #|   > 0 - retry up to X times

            #'POLL_READ_LATENCY':(Optional[float|int]), # Grace time in seconds for␣
↪receiving the reply from
                                    #server. Will be added to the `timeout` value and␣
↪used as the read timeout from
                            #server (Default: 2).
        },
        #Other bots here with same structure.
    ],

}
```

Include in your urls.py the `django_telegrambot.urls` (NB: If you upgrade from a previous version, you MUST change how to include `django_telegrambot.urls`. Never set prefix here!):

```
#urls.py
urlpatterns = [
    ...
    url(r'^', include('django_telegrambot.urls')),
    ...
]
```

Then use it in a project creating a module `telegrambot.py` in your app

```
#myapp/telegrambot.py
# Example code for telegrambot.py module
from telegram.ext import CommandHandler, MessageHandler, Filters
from django_telegrambot.apps import DjangoTelegramBot

import logging
logger = logging.getLogger(__name__)


# Define a few command handlers. These usually take the two arguments bot and
# update. Error handlers also receive the raised TelegramError object in error.
def start(bot, update):
    bot.sendMessage(update.message.chat_id, text='Hi!')


def help(bot, update):
    bot.sendMessage(update.message.chat_id, text='Help!')
```

```
def echo(bot, update):
    bot.sendMessage(update.message.chat_id, text=update.message.text)


def error(bot, update, error):
    logger.warn('Update "%s" caused error "%s"' % (update, error))


def main():
    logger.info("Loading handlers for telegram bot")

    # Default dispatcher (this is related to the first bot in settings.DJANGO_
↪TELEGRAMBOT['BOTS'])
    dp = DjangoTelegramBot.dispatcher
    # To get Dispatcher related to a specific bot
    # dp = DjangoTelegramBot.getDispatcher('BOT_n_token')     #get by bot token
    # dp = DjangoTelegramBot.getDispatcher('BOT_n_username')  #get by bot username

    # on different commands - answer in Telegram
    dp.add_handler(CommandHandler("start", start))
    dp.add_handler(CommandHandler("help", help))

    # on noncommand i.e message - echo the message on Telegram
    dp.add_handler(MessageHandler([Filters.text], echo))

    # log all errors
    dp.add_error_handler(error)
```

Features

- Multiple bots
- Admin dashboard available at `/admin/django-telegrambot`
- Polling mode by management command (an easy to way to run bot in local machine, not recommended in production!)

  ```
  (myenv) $ python manage.py botpolling --username=<username_bot>
  ```

Contributing

Patches and bug reports are welcome, just please keep the style consistent with the original source.

Running Tests

Does the code actually work?

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install -r requirements-test.txt
(myenv) $ python runtests.py
```

Sample Application

There a sample application in *sampleproject* directory. Here is installation instructions:

1. Install requirements with command

   pip install -r requirements.txt

2. Copy file *local_settings.sample.py* as *local_settings.py* and edit your bot token

   cp sampleproject/local_settings.sample.py sampleproject/local_settings.py

   nano sampleproject/local_settings.py

3. Run Django migrations

   python manage.py migrate

4. Run server

   python manage.py runserver

5. If **WEBHOOK** Mode setted go to 8
6. If **POLLING** Mode setted, open in your browser http://localhost/
7. Open Django-Telegram Dashboard http://localhost/admin/django-telegrambot and follow instruction to run worker by management command *botpolling*. Then go to 10
8. To test webhook locally install *ngrok* application and run command

   ./ngrok http 8000

9. Change *WEBHOOK_SITE* and *ALLOWED_HOSTS* in local_settings.py file
10. Start a chat with your bot using telegram.me link avaible in **Django-Telegram Dashboard** at http://localhost/admin/django-telegrambot

Credits

Required package:

- Python Telegram Bot

Tools used in rendering this package:

- Cookiecutter

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-telegrambot
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-telegrambot
$ pip install django-telegrambot
```

CHAPTER 3

Usage

To use django-telegrambot in a app, create a telegrambot.py module in your app as follow:

```
# Example code for telegrambot.py module
from telegram.ext import CommandHandler, MessageHandler, Filters
from django_telegrambot.apps import DjangoTelegramBot

import logging
logger = logging.getLogger(__name__)


# Define a few command handlers. These usually take the two arguments bot and
# update. Error handlers also receive the raised TelegramError object in error.
def start(bot, update):
    bot.sendMessage(update.message.chat_id, text='Hi!')


def help(bot, update):
    bot.sendMessage(update.message.chat_id, text='Help!')


def echo(bot, update):
    bot.sendMessage(update.message.chat_id, text=update.message.text)


def error(bot, update, error):
    logger.warn('Update "%s" caused error "%s"' % (update, error))


def main():
    logger.info("Loading handlers for telegram bot")

    # Default dispatcher (this is related to the first bot in settings.DJANGO_
↪TELEGRAMBOT['BOTS'])
    dp = DjangoTelegramBot.dispatcher
    # To get Dispatcher related to a specific bot
```

```
    # dp = DjangoTelegramBot.getDispatcher('BOT_n_token')     #get by bot token
    # dp = DjangoTelegramBot.getDispatcher('BOT_n_username')  #get by bot username

    # on different commands - answer in Telegram
    dp.add_handler(CommandHandler("start", start))
    dp.add_handler(CommandHandler("help", help))

    # on noncommand i.e message - echo the message on Telegram
    dp.add_handler(MessageHandler([Filters.text], echo))

    # log all errors
    dp.add_error_handler(error)
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/JungDev/django-telegrambot/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.
- Detailed steps to reproduce the bug.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/JungDev/django-telegrambot/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 :)

Get Started!

Ready to contribute? Here's how to set up *django-telegrambot* for local development.

1. Fork the *django-telegrambot* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-telegrambot.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-telegrambot
$ cd django-telegrambot/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 django_telegrambot tests
$ python setup.py test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/JungDev/django-telegrambot/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest tests.test_django_telegrambot
```

CHAPTER 5

Credits

Development Lead

- django-telegrambot <francesco.scarlato@gmail.com>

Contributors

None yet. Why not be the first?

CHAPTER 6

History

1.0.0 (2017-05-25)

- IMPORTANT: If you upgrade from a previous version, you MUST change how to include django_telegrambot.urls and settings.py.
- Added admin dashboard, available at /admin/django-telegrambot
- Added polling mode from management command (an easy to way to run bot in local machine, not recommended in production)
- More setting available
- Improved AppConfig
- Improved sample project

0.2.6 (2017-04-08)

- Improved module loading
- Added sample project

0.2.5 (2017-03-06)

- Fix compatibility with python-telegram-bot 5.1

0.2.4 (2016-10-04)

- Fix compatibility with Django 1.10

0.2.3 (2016-07-30)

- Fix default dispatcher and bot

0.2.2 (2016-07-27)

- Fix multi workers

0.2.1 (2016-07-24)

- Update for python-telegram-bot release v5.0

0.2.0 (2016-04-27)

- Update for python-telegram-bot release v4.0.1

0.1.8 (2016-03-22)

- Update for deprecation in python-telegram-bot release v3.4

0.1.5 (2016-01-28)

- Fix compatibility.

0.1.4 (2016-01-28)

- Fix compatibility.

0.1.3 (2016-01-28)

- Fix setting certificate.
- Add method DjangoTelegramBot.getBot(); get bot instance by token or username.

0.1.2 (2016-01-26)

- First release on PyPI.