twodolib Documentation

Release 0.5.4

Karsten Schulz

Nov 27, 2019

Contents

Contents:

CHAPTER 1

twodolib - A commandline helper to add tasks to 2DoApp

1.1 Description

This package provides the library `twodolib` and a command line utility `task2do` to add tasks, projects and checklists to the macOS App 2DoApp from the command line.

Since version 1.5 (Mac) 2Do supports adding tasks by using an URL scheme. For example, if you want to add the task *Save the world.*, you can open the URL:

```
twodo://x-callback-url/add?task=Save%20the%20world.
```

to add this task to your 2Do App (see: https://www.2doapp.com/kb/article/url-schemes.html)

The `task2do` command supports creating such URLs from the command line. To print such an URL for a task without adding it, just enter:

```
task2do "Save the world."
```

which prints the URL to stdout like this:

```
twodo://x-callback-url/add?task=Save%20the%20world.
```

If you want to actually add the task to your 2Do App, use the `-e` or `--execute` option:

```
task2do -e "Save the world."
# no output here, but the task should be added into your standard list in 2Do
```

1.2 Features

- runs with Python 3 (if you need py27 support please use release 0.4.0)
- Create tasks on the command line and show the corresponding URL scheme, for copy and pasting it.
- Create tasks on the command line and open the corresponding URL scheme to send it to 2DoApp

See the documentation at http://twodolib.readthedocs.org/en/latest/

1.3 Install

See docs/installation.rst (It's just `pip install twodolib`)

1.4 Dependencies

- wheel

1.5 License

- Free software: ISC license

CHAPTER 2

Installation

At the command line:

```
$ pip install twodolib
# or
$ pip install --user twodolib
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv twodolib
(twodolib)$ pip install twodolib
```

CHAPTER 3

Usage

3.1 Using twodolib as a Python module

Create a task:

```
>>> from twodolib import TwoDoTask
>>> my_task = TwoDoTask('Save the world.', priority=TwoDoTask.PRIO_HIGH)

>>> my_task.url()
u'twodo://x-callback-url/add?task=Save%20the%20world.&priority=3'
```

Create a task, save it to 2DoApp and retrieve the task id:

```
>>> my_task = TwoDoTask('Save the world.', priority=TwoDoTask.PRIO_HIGH, for_
↪list="inbox")
```

```
>>> # save the task in 2DoApp
>>> import subprocess
>>> subprocess.call(['open', my_task.url()])
0
```

```
>>> # retrieve taskid
>>> my_task.get_taskid()
```

'181378c217dd4f029a6ec7a47e65550d'

3.2 Using twodolib from the command line

You can call `task2do` with a whole bunch of options. But it is easy to add a standard task. Just enter:

```
task2do -d 1 "Dinner at 8pm."
```

And you will get the URL to add a task, which is due tomorrow:

```
twodo://x-callback-url/add?task=Dinner%20at%208pm.&due=1
```

Use the URL with Safari or Alfred or another launcher to actually create the task. Or just provide the `-e, --execute` option, to actually create the task in your 2Do App:

```
task2do -e -d 1 "Dinner at 8pm."
```

You'll get help with the `-h, --help` option:

```
usage: task2do [-h] [-a ACTION] [-d DUE] [--dueTime DUETIME] [-e]
               [-f FOR_PARENTNAME] [-i] [-l FOR_LIST] [-n NOTE] [-p {0,1,2,3}]
               [--project IN_PROJECT] [--repeat {1,2,3,4}] [-s]
               [--start START] [-t {0,1,2}] [--tags TAGS] [--taskid] [-v]
               task

Program to create tasks in 2Do. The default behavior is to print the generated
URL to stdout. Please use the '-e' or '--execute'' option, if you want to send
the task directly to the 2DoApp.

Examples
========

Add a task due tomorrow:
task2do -d 1 "Dinner at 8pm."

Add a task with high priority:
task2do -p 3 "High priority task."
- or -
task2do --priority 3 "High priority task."

Add a task due today and repeated weekly:
task2do "change clothes" -d 0 --repeat 2

Add a task due at 6pm today
task2do "Watch EX_MACHINA" --due 0 --dueTime 18:00

Add a task due tomorrow, with tags, which is also starred and repeated monthly
task2do "Monthly subscription." --tags bill,payment -s --due 1 --repeat 4

Add a task with an url action (open a link)
task2do "Read help text" -a url:https://www.2doapp.com/
task2do "Read help text" --action url:https://www.2doapp.com/

Add a Subtask in list personal in project errands:
task2do "Buy milk." -l personal --project errands

positional arguments:
  task                  Title of the task.

optional arguments:
  -h, --help            show this help message and exit
  -a ACTION, --action ACTION
                        action: Supports the following formats: call:<number>
                        - Add a Call action to call the specified number
                        message:<number> - Add a Message action to send a
                        message to the specified number mail:<email> - Add a
```

(continues on next page)

(continued from previous page)

```
                        Email action to send an email the specified email
                        address url:<url to visit> - Add a Browse action to
                        visit the specified URL address visit:<address> - Add
                        a Visit action to visit the specified location
                        google:<search term> - Add a Google action to search
                        the specified keyword Enter the arguments after the
                        colon without the angle brackets. For more details:
                        SEE https://www.2doapp.com/kb/article/url-schemes.html
  -d DUE, --due DUE     Due date. Supports two formats: YYYY-MM-DD - Sets the
                        date on default due time (based on your settings),
                        unless due time is specified separately or
                        ignoreDefaults (-i) is given. OR: Number of days due
                        from today. Which means: 0 = today, 1 = tomorrow and
                        so on)
  --dueTime DUETIME     Due time. Supports 24h format HH:MM.
  -e, --execute         Actually add the task instead of only printing the URL
                        to stdout.
  -f FOR_PARENTNAME, --forParentName FOR_PARENTNAME
                        Title of an existing project or checklist to save the
                        new task there as a subtask. Also requires the
                        parent's task list.
  -i, --ignoreDefaults  Ignore default date / time settings of 2DoApp.
  -l FOR_LIST, --list FOR_LIST
                        Name of an existing list in 2DoApp, case-insensitive.
                        If missing, the default list or the currently visible
                        list on screen is used.
  -n NOTE, --note NOTE  Notes for the task
  -p {0,1,2,3}, --priority {0,1,2,3}
                        priority: 0 (none), 1 (low), 2 (medium), 3 (high)
  --project IN_PROJECT  Name of an existing project in 2DoApp, into which the
                        task will be pasted. So you can create subtasks.
  --repeat {1,2,3,4}    Repeat task: 1 (daily), 2 (weekly), 3 (bi-weekly), 4
                        (monthly))
  -s, --starred         Mark task as starred.
  --start START         Start date and time. Supports the format: "YYYY-MM-DD
                        HH:MM" - Sets the start date to the date and time
                        specified - OR - Any number with 0 = today, 1 =
                        tomorrow and so on)
  -t {0,1,2}, --type {0,1,2}
                        Type of task to create. The following options are
                        supported: 0 - Task (default), 1 - Project, 2 -
                        Checklist
  --tags TAGS           Comma separated list of tags to assign to the task
  --taskid              Prints taskid, needs the task title and the list.
  -v, --version         show program's version number and exit
```

3.3 Examples

Add a task due Tomorrow:

```
task2do -d 1 "Dinner at 8pm."
```

Add a task with high priority:

```
task2do -p 3 "High priority task."
task2do --priority 3 "High priority task."
```

Add a task due today and repeated weekly:

```
task2do "change clothes" -d 0 --repeat 2
```

Add a task due at 6pm today:

```
task2do "Watch EX_MACHINA" --due 0 --dueTime 18:00
```

Add a task due tomorrow, with tags, which is also starred and repeated monthly:

```
task2do "Monthly subscription." --tags bill,payment -s --due 1 --repeat 4
```

Add a Subtask in list business in project webpage:

```
task2do "Make webpage GDPR compatible." -l business --project webpage
```

If you use a project, you must provide a list, too.

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/KarstenSchulz/twodolib/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/KarstenSchulz/twodolib/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 :)

4.2 Get Started!

Ready to contribute? Here's how to set up *twodolib* for local development.

1. Fork the *twodolib* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/twodolib.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 twodolib
$ cd twodolib/
$ 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 twodolib 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.

4.3 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, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/KarstenSchulz/twodolib/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

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

CHAPTER 5

Credits

5.1 Development Lead

- Karsten Schulz <github@karstenschulz.biz>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.5.4 (2019-07-28)

updated HISTORY.txt file (gosh!)

6.2 0.5.3 (2019-07-28)

removes unnecessary dependencies.

6.3 0.5.2 (2019-07-15)

Updated requirements.

6.4 0.5.1 (2018-11-14)

Please update, because package requests was updated for security reasons!

- updated requirements

6.5 0.5.0 (2018-08-08)

- dropping Python 2.7 support
- implemented `forParentName`
- print taskid

6.6 0.4.0 (2018-08-07)

- you can paste tasks now, even as subtasks of a project. See help or docs!
- minor fixes

6.7 0.3.0 (2017-05-28)

- added 'action' property of tasks (e.g. url, phone, . . .)
- switched from webbrowser.open to subprocess.call(['open', . . .)

6.8 0.2.1 (2015-09-21)

- fixed classifier in setup.py
- fixed #3 - adding to lists works now.

6.9 0.2.0 (2015-09-20)

- first public release.

6.10 0.1.0 (unreleased)

- ADD: more commandline options implemented: repeat, due, dueTime, start
- ADD: docs

6.11 0.0.1 (2015-09-14)

- not released - pre alpha.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search