
tasks_collector

Daniel Engvall

Nov 29, 2019

CONTENTS:

1	README Tasks Collector	1
1.1	Background	1
1.2	Requirements	1
1.3	Introduction	1
1.4	Installation	1
1.5	Usage graphical interface	1
1.6	Usage command-line	2
1.7	Create sphinx documentation	4
1.8	Test the code with coverage	4
1.9	API documentation	4
1.10	Troubleshooting	5
1.11	Contact	5
2	tasks_collector	7
2.1	tasks_collector package	7
3	Indices and tables	13
4	Most recent changes	15
	Python Module Index	17
	Index	19

README TASKS COLLECTOR

1.1 Background

The purpose with this project was to address the headache of collecting and organizing the tasks are/have worked with. These tasks spread across different platforms such as Outlook, Jira, Trello and other platforms and felt like I had to structure and store these into a database and being able to create some charts based on this.

1.2 Requirements

At this moment this application has yet only been tested from MacOS High Sierra and above. Some parts of the application related to Outlook (through AppleScript) and CopyQ only available to MacOS.

1.3 Introduction

This below is a brief run through how one could use this tool from your terminal and how you can use this package in your own Python applications.

1.4 Installation

```
bash-4.4$ python -m venv venv
bash-4.4$ source venv/bin/activate
(venv) bash-4.4$ pip install tasks-collector
```

1.5 Usage graphical interface

The default entry-point for this application is GUI (based on the great module Gooey) to simplify building a window presenting all options.

1.6 Usage command-line

To use command-line you need always supply the **–ignore-gooey** flag to disable GUI (graphical interface) In general you only need to pass the sqlite database where you'd like to store it together with the flag depending which source to use

```
$ tasks_collector collect --help --ignore-gooey
usage: tasks_collector collect [-h] [--outlook] [--jira JIRA]
                               [--trello TRELLO]
                               [--sqlite_database SQLITE_DATABASE]
                               [--loglevel {INFO,DEBUG}]

optional arguments:
  -h, --help            show this help message and exit
  --outlook
  --jira JIRA            username@jiraserver
  --trello TRELLO        api_key:token:token_secret:my_name
  --sqlite_database SQLITE_DATABASE
                        name of sqlite to export/update to
  --loglevel {INFO,DEBUG}
```

1.6.1 Collect

Outlook

When passing **–outlook** argument you just need to make sure you've selected all Outlook-tasks including those completed. While using Outlook you can add the following naming convention of Outlooks "Categories"

```
(client1)
(client2)
(client3)
...
```

for giving you possibility to assign clients associated with task.

```
{project1}
{project2}
{project3}
...
```

for assigning the task specific project.

Jira

The script will use the `username@jiraserver` supplied to detect all tasks that are assigned to you and collect their most recent details into the database. Name of the board will become the representation of "client"

Trello

The script will use an argument structured as `'api-key:token:token_secret:my_name'` You will get api-key, token and token secret from `https://trello.com/app-key` The `my_name` is the name as your logged in user has, this is to be able to identify "your" tasks amongst others in boards. The board name will be the representation of "client"

Credentials

Currently use keyring to allow you to store credentials locally not being exposed. First time you run it will prompt you for password.

Report

Install CopyQ according to <https://hluk.github.io/CopyQ/> if you like to be able to export your report to your clipboard using the `--copyq` flag. But you can also get the graph when using the `--show`

```
$ tasks_collector report --help --ignore-gooey
2019-11-25 18:46:09.199 | INFO      | tasks_collector.__main__:main:79 - no gui
2019-11-25 18:46:09.200 | DEBUG    | tasks_collector.tasksdb.api:get_default_db_
↳path:48 - no local config directory, directory will be used /Users/edo/Library/
↳Application Support/tasks_collector.tasksdb
usage: tasks_collector report [-h] [--days number_of_days_in_past]
                             [--sqlite_database SQLITE_DATABASE] [--copyq]
                             [--show]
                             [--default_client name of default client]
                             [--loglevel {INFO,DEBUG}]

optional arguments:
  -h, --help            show this help message and exit
  --days number_of_days_in_past
                        Number of days to cover in the report
  --sqlite_database SQLITE_DATABASE
                        name of sqlite to export/update to
  --copyq               paste output as MIME to pastebin, good for sending by
                        e-mail
  --show                show gantt image
  --default_client name of default client
  --loglevel {INFO,DEBUG}
```

A sample of the output after pasting into e.g. a email for you manager might look like this

Cleanup

This is useful if for example ownership changes of tickets in Jira end you'd like to close them in your report.

```
$ tasks_collector cleanup --help --ignore-gooey
usage: tasks_collector cleanup [-h] [--before DAYS]
                               [--sqlite_database SQLITE_DATABASE]
                               [--loglevel {INFO,DEBUG}]

optional arguments:
  -h, --help            show this help message and exit
  --before DAYS         tickets before this number of days back to be closed
  --sqlite_database SQLITE_DATABASE
                        name of sqlite to export/update to
  --loglevel {INFO,DEBUG}
```

1.7 Create sphinx documentation

To create documentation in docs/build/html

```
$ cd docs
$ sphinx-apidoc -o build/ ../tasks_collector
$ make html
```

1.8 Test the code with coverage

Thanks to tox this has been automated so all that you need to run “tox” from projects root directory e.g.

```
$ tox
...
py3.7.0 run-test: commands[0] | pytest --cov=tasks_collector tests/
.....
↪
↪ [100%]
----- coverage: platform darwin, python 3.7.1-final-0 -----
Name                               Stmts  Miss  Cover
-----
tasks_collector/__init__.py          1     0   100%
tasks_collector/reportgenerator/__init__.py  0     0   100%
tasks_collector/reportgenerator/api.py    234   117    50%
tasks_collector/tasksconverter/__init__.py  0     0   100%
tasks_collector/tasksconverter/api.py    100     6    94%
tasks_collector/tasksdb/__init__.py       0     0   100%
tasks_collector/tasksdb/api.py          80     8    90%
tasks_collector/tasksscraeper/__init__.py  0     0   100%
tasks_collector/tasksscraeper/jirascraeper.py  17     3    82%
tasks_collector/tasksscraeper/outlookscraeper.py  17     1    94%
tasks_collector/tasksscraeper/trelloscraeper.py  37    13    65%
-----
TOTAL                                486   148    70%

27 passed, 4 warnings in 19.63s

↪ _____ summary _____
↪
py3.4.0: commands succeeded
py3.5.6: commands succeeded
py3.6.6: commands succeeded
py3.7.0: commands succeeded
congratulations :)
```

1.9 API documentation

Could be found at <https://tasks-collector.readthedocs.io>

1.10 Troubleshooting

I get the following message when I run tasks_collector in GUI (not `--ignore-gooey`) mode

```
This program needs access to the screen. Please run with a  
Framework build of python, and only when you are logged in  
on the main display of your Mac.
```

Reasons for this happening on MacOS is that your built of Python3 does not include “framework” (e.g. when you installed through homebrew) rather than from <https://www.python.org/downloads/mac-osx/>

One way around this if you’d be using pyenv (one of my favourites) found at <https://github.com/pyenv/pyenv> is to install Python using the following

```
env PYTHON_CONFIGURE_OPTS="--enable-framework CC=clang" pyenv install 3.7.0
```

1.11 Contact

You can easiest contact me by my email daniel@engvalls.eu or my linked-in profile <https://www.linkedin.com/in/danielengvall/>

TASKS_COLLECTOR

2.1 tasks_collector package

2.1.1 Subpackages

tasks_collector.reportgenerator package

Submodules

tasks_collector.reportgenerator.api module

tasksconverter:

tasks_collector.reportgenerator.api.**all_values** (*in_data: List*, *key: str = 'name'*) → List
Create sorted list

Args: in_data: key:

Returns:

tasks_collector.reportgenerator.api.**count_items** (*in_data: List*, *dict_condition: Dict*) → int
Count items

Args: in_data: dict_condition:

Returns:

tasks_collector.reportgenerator.api.**create_concurrent_chart** (*concurrent_list*,
date_key='date',
show_plot=False,
concurrent_file='/tmp/concurrent.png',
dpi=72)

tasks_collector.reportgenerator.api.**create_concurrent_list** (*in_data: List*,
name_key: str = 'name', *start_key: str = 'start_date'*,
end_key: str = 'close_date') → List
Creates list of concurrent tasks to generate plot of

Args: in_data: name_key: start_key: end_key:

Returns: List to generate plots

`tasks_collector.reportgenerator.api.create_gantt_chart` (*task_list: List, *, show_plot: bool = True, gantt_file: str = '/tmp/gantt.png', dpi: int = 72*) → str

Creates gantt chart

Args: task_list: show_plot: gantt_file: dpi:

Returns: A base64 encoded image

`tasks_collector.reportgenerator.api.create_gantt_list` (*generic_tasks: List, default_client='None'*) → List

Creates a list formatting tasks names etc

Args: default_client: generic_tasks:

Returns: New list

`tasks_collector.reportgenerator.api.dict_keys_to_ymd` (*_dict: Dict, _keys: List = []*) → Dict

Update keys with dates

Args: _dict: _keys:

Returns: New dict with parsed dates

`tasks_collector.reportgenerator.api.filter_generic_tasks` (*generic_task_list: List, *, from_date: Optional[str] = None, to_date: Optional[str] = None, include_open: bool = True*) → List

Filter generic tasks based on criterias

Args: generic_task_list: from_date: to_date: include_open:

Returns: A new task list

`tasks_collector.reportgenerator.api.get_gantt_b64` (*gantt_list: List, show_gantt: bool = True*) → str

Returns base64 encoded string of gantt

Args: gantt_list: show_gantt:

Returns:

`tasks_collector.reportgenerator.api.get_lowest_value` (*input_dict_list: Dict, key_name: str*) → int

Get lowest value

Args: input_dict_list: key_name:

Returns:

`tasks_collector.reportgenerator.api.render_task` (*client: str, category: str, subject: str, **kwargs*) → str

Render HTML for task

Args: client: category: subject:

Returns:

```
tasks_collector.reportgenerator.api.tasks_to_pastebin(generic_tasks:      List,  
                                                    _filter:    bool = False,  
                                                    show_gantt: bool = True,  
                                                    default_client='None') →  
                                                    None
```

Creates tasks and inserted to pastebin

Args: default_client: generic_tasks: _filter: show_gantt:

Returns:

Module contents

tasks_collector.tasksconverter package

Submodules

tasks_collector.tasksconverter.api module

tasksconverter:

```
tasks_collector.tasksconverter.api.convert_date_attribute(date:  
                                                         Union[datetime.datetime,  
                                                         str]) → Optional[str]
```

Converting date attribute

Args: date:

Returns:

```
tasks_collector.tasksconverter.api.correct_task(input_dict: Dict) → Union[str, Dict]  
Correcting the task
```

Args: input_dict:

Returns:

```
tasks_collector.tasksconverter.api.format_subject(subject: str, _type='outlook') → str  
Properly format subject
```

Args: subject: _type:

Returns:

```
tasks_collector.tasksconverter.api.parse_category(category_list:      List,  
                                                  _type='outlook') → Dict
```

Parse categories

Args: category_list: _type:

Returns:

```
tasks_collector.tasksconverter.api.to_generic(tasks_list: List, _type='outlook') → List  
Make list of tasks generic
```

Args: tasks_list: _type:

Returns: New list of generic tasks

Module contents

tasks_collector.tasksdb package

Submodules

tasks_collector.tasksdb.api module

tasksdb: ...

```
class tasks_collector.tasksdb.api.BaseModel(*args, **kwargs)
    Bases: peewee.Model
```

```
    DoesNotExist
        alias of BaseModelDoesNotExist
```

```
    id = <AutoField: BaseModel.id>
```

```
class tasks_collector.tasksdb.api.OpenDB(db_file, _type='sqlite')
    Bases: object
```

A class to simplify creation of a database.

```
    static get_all_tasks() → List
```

```
class tasks_collector.tasksdb.api.Task(*args, **kwargs)
    Bases: tasks_collector.tasksdb.api.BaseModel
```

```
    DoesNotExist
        alias of TaskDoesNotExist
```

```
    category = <CharField: Task.category>
```

```
    client = <CharField: Task.client>
```

```
    close_date = <DateField: Task.close_date>
```

```
    due_date = <DateField: Task.due_date>
```

```
    id = <AutoField: Task.id>
```

```
    modified_date = <DateField: Task.modified_date>
```

```
    start_date = <DateField: Task.start_date>
```

```
    status = <CharField: Task.status>
```

```
    subject = <CharField: Task.subject>
```

```
tasks_collector.tasksdb.api.cleanup(before_date)
    Cleanup the database
```

Args: before_date:

Returns: None

```
tasks_collector.tasksdb.api.get_default_db_path() → str
    Get a default database path
```

Returns: str: path

```
tasks_collector.tasksdb.api.get_kv_task_as_text(task: tasks_collector.tasksdb.api.Task,
        remove_keys: List = ['id']) → Dict
```

tasks_collector.tasksdb.api.**insert_or_updates_tasks** (*tasks_list_dict: List*) → None
Insert or update task

Args: tasks_list_dict: List of tasks to update

Returns: None

Module contents

tasks_collector.tasksscraeper package

Submodules

tasks_collector.tasksscraeper.jirascraper module

tasksscraeper.jira: ...

tasks_collector.tasksscraeper.jirascraper.**get_jira_tasks** (*host: str, username: str, jira_password: str, max_results: int = 1000*) → List

Query Jira for tickets

Args: host: username: jira_password: max_results:

Returns: List of tasks

tasks_collector.tasksscraeper.outlookscraper module

tasksscraeper.outlook: ...

tasks_collector.tasksscraeper.outlookscraper.**get_outlook_tasks** () → List
Get Outlook tasks

Returns: List of tasks from Outlook

tasks_collector.tasksscraeper.trelloscraper module

tasksscraeper.trello: ...

tasks_collector.tasksscraeper.trelloscraper.**add_project** (*input_my_tasks, board_ids*)
tasks_collector.tasksscraeper.trelloscraper.**find_my_id** (*input_client, input_name*)
tasks_collector.tasksscraeper.trelloscraper.**get_all_board_ids** (*input_client, input_tasks*)
tasks_collector.tasksscraeper.trelloscraper.**get_trello_tasks** (*api_key: str, token: str, token_secret: str, my_name: str*) → List

Query Trello for tickets

Args: my_name: api_key: token: token_secret:

Returns: List of tasks

Module contents

2.1.2 Module contents

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

MOST RECENT CHANGES

- **Adding new installation instructions based on pip install from PyPI** by *Daniel Engvall* at 2019-11-29 20:33:04
- **Adding troubleshooting section** by *Daniel Engvall* at 2019-11-29 20:23:46
- **Bump to version 0.9.1** by *Daniel Engvall* at 2019-11-29 19:54:40
- **Removing applescript module no longer required** by *Daniel Engvall* at 2019-11-29 19:47:12
- **Updating classifiers in setup.py** by *Daniel Engvall* at 2019-11-29 19:03:01
- **Updating travis-ci yaml adding badge for code coverage** by *Daniel Engvall* at 2019-11-29 18:59:32
- **Updating travis-ci yaml adding badge for code coverage** by *Daniel Engvall* at 2019-11-29 18:56:31
- **Updating travis-ci yaml adding codecov.io** by *Daniel Engvall* at 2019-11-29 18:33:03
- **Updating travis-ci yaml adding codecov.io** by *Daniel Engvall* at 2019-11-29 18:27:47
- **Updating travis-ci yaml using osx instead** by *Daniel Engvall* at 2019-11-29 17:56:05

PYTHON MODULE INDEX

t

- tasks_collector, [12](#)
- tasks_collector.reportgenerator, [9](#)
- tasks_collector.reportgenerator.api, [7](#)
- tasks_collector.tasksconverter, [10](#)
- tasks_collector.tasksconverter.api, [9](#)
- tasks_collector.tasksdb, [11](#)
- tasks_collector.tasksdb.api, [10](#)
- tasks_collector.tasksscrafer, [12](#)
- tasks_collector.tasksscrafer.jirascraper,
 [11](#)
- tasks_collector.tasksscrafer.outlookscraper,
 [11](#)
- tasks_collector.tasksscrafer.trelloscraper,
 [11](#)

A

`add_project()` (in module `tasks_collector.tasksscrafer.trelloscraper`), 11

`all_values()` (in module `tasks_collector.reportgenerator.api`), 7

B

`BaseModel` (class in `tasks_collector.tasksdb.api`), 10

C

`category` (`tasks_collector.tasksdb.api.Task` attribute), 10

`cleanup()` (in module `tasks_collector.tasksdb.api`), 10

`client` (`tasks_collector.tasksdb.api.Task` attribute), 10

`close_date` (`tasks_collector.tasksdb.api.Task` attribute), 10

`convert_date_attribute()` (in module `tasks_collector.tasksconverter.api`), 9

`correct_task()` (in module `tasks_collector.tasksconverter.api`), 9

`count_items()` (in module `tasks_collector.reportgenerator.api`), 7

`create_concurrent_chart()` (in module `tasks_collector.reportgenerator.api`), 7

`create_concurrent_list()` (in module `tasks_collector.reportgenerator.api`), 7

`create_gantt_chart()` (in module `tasks_collector.reportgenerator.api`), 8

`create_gantt_list()` (in module `tasks_collector.reportgenerator.api`), 8

D

`dict_keys_to_ymd()` (in module `tasks_collector.reportgenerator.api`), 8

`DoesNotExist` (`tasks_collector.tasksdb.api.BaseModel` attribute), 10

`DoesNotExist` (`tasks_collector.tasksdb.api.Task` attribute), 10

`due_date` (`tasks_collector.tasksdb.api.Task` attribute), 10

F

`filter_generic_tasks()` (in module `tasks_collector.reportgenerator.api`), 8

`find_my_id()` (in module `tasks_collector.tasksscrafer.trelloscraper`), 11

`format_subject()` (in module `tasks_collector.tasksconverter.api`), 9

G

`get_all_board_ids()` (in module `tasks_collector.tasksscrafer.trelloscraper`), 11

`get_all_tasks()` (`tasks_collector.tasksdb.api.OpenDB` static method), 10

`get_default_db_path()` (in module `tasks_collector.tasksdb.api`), 10

`get_gantt_b64()` (in module `tasks_collector.reportgenerator.api`), 8

`get_jira_tasks()` (in module `tasks_collector.tasksscrafer.jirascraper`), 11

`get_kv_task_as_text()` (in module `tasks_collector.tasksdb.api`), 10

`get_lowest_value()` (in module `tasks_collector.reportgenerator.api`), 8

`get_outlook_tasks()` (in module `tasks_collector.tasksscrafer.outlookscraper`), 11

`get_trello_tasks()` (in module `tasks_collector.tasksscrafer.trelloscraper`), 11

I

`id` (`tasks_collector.tasksdb.api.BaseModel` attribute), 10

`id` (`tasks_collector.tasksdb.api.Task` attribute), 10

`insert_or_updates_tasks()` (in module `tasks_collector.tasksdb.api`), 10

M

`modified_date` (`tasks_collector.tasksdb.api.Task` attribute), 10

O

OpenDB (*class in tasks_collector.tasksdb.api*), 10

P

parse_category() (*in module tasks_collector.tasksconverter.api*), 9

R

render_task() (*in module tasks_collector.reportgenerator.api*), 8

S

start_date (*tasks_collector.tasksdb.api.Task attribute*), 10

status (*tasks_collector.tasksdb.api.Task attribute*), 10

subject (*tasks_collector.tasksdb.api.Task attribute*), 10

T

Task (*class in tasks_collector.tasksdb.api*), 10

tasks_collector (*module*), 12

tasks_collector.reportgenerator (*module*), 9

tasks_collector.reportgenerator.api (*module*), 7

tasks_collector.tasksconverter (*module*), 10

tasks_collector.tasksconverter.api (*module*), 9

tasks_collector.tasksdb (*module*), 11

tasks_collector.tasksdb.api (*module*), 10

tasks_collector.tasksscraeper (*module*), 12

tasks_collector.tasksscraeper.jirascraper (*module*), 11

tasks_collector.tasksscraeper.outlookscraper (*module*), 11

tasks_collector.tasksscraeper.trelloscraper (*module*), 11

tasks_to_pastebin() (*in module tasks_collector.reportgenerator.api*), 8

to_generic() (*in module tasks_collector.tasksconverter.api*), 9