Todoman Documentation

Release 2.0.2

Hugo Osvaldo Barrera

February 02, 2017

Contents

1 Features 3

2 Contributing 5

3 Caveats 7

4 Table of Contents 9
- 4.1 Installing . . . 9
- 4.2 Configuring . . . 9
- 4.3 Usage . . . 10
- 4.4 Contributing . . . 12
- 4.5 Changelog . . . 13
- 4.6 Licence . . . 13

5 Indices and tables 15

Todoman is a simple, standards-based, cli todo (aka: task) manager. Todos are stored into icalendar files, which means you can sync them via CalDAV using, for example, vdirsyncer.

Todoman is now part of the `pimutils` project, and is hosted at GitHub. The original location at GitLab.com is still kept as a mirror.

CHAPTER 1

Features

- Listing, editing and creating todos.
- Todos are read from individual ics files from the configured directory. This matches the vdir specification.
- There's support for the most common TODO features for now (summary, description, location, due date and priority) for now.
- Todoman should run on any major operating system.
- Unsupported fields may not be shown but are *never* deleted or altered.

CHAPTER 2

Contributing

See Contributing for details on contributing.

CHAPTER 3

Caveats

Priority granularity hasn't been completely implemented yet. Icalendar supports priorities 1-9 or none. Todoman supports only none or 1 (highest).

Due dates are generally shown and editable as dates with no time component.

Support for the `percent-completed` attribute is incomplete. Todoman can only mark todos as completed (100%), and will nor reflect nor allow editing for values for `percent > 0 ^ percent < 100`.

CHAPTER 4

Table of Contents

4.1 Installing

If todoman is packaged for your OS/distribution, using your system's standard package manager is probably the easiest way to install khal:

- ArchLinux (AUR)

4.1.1 Install via PIP

Since *todoman* is written in python, you can use python's package managers, *pip* by executing:

```
pip install todoman
```

or the latest development version by executing:

```
pip install git+git://github.com/pimutils/todoman.git
```

This should also take care of installing all required dependencies.

4.1.2 Manual installation

If pip is not available either (this is most unlikely), you'll need to download the source tarball and install via setup.py, though this is not a recomended installation method:

```
python3 setup.py install
```

4.1.3 Requirements

Todoman requires python 3.3 or later. Installation of required libraries can be done via pip, or your OS's package manager. If you're interested in packaging todoman, all depenencies are listed in requirements.txt.

Todoman will not work with python 2. However, keep in mind that python 2 and python 3 can coexist (and most distributions actually ship both).

4.2 Configuring

You'll need to configure Todoman before the first usage, using its simple ini-like configuration file.

4.2.1 Configuration File

The configuration file should be placed in `$XDG_CONFIG_DIR/todoman/todoman.conf`. `$XDG_CONFIG_DIR` defaults to `~/.config` is most situations, so this will generally be `~/.config/todoman/todoman.conf`.

Main section

- `path`: A glob pattern matching the directories where your todos are located.
- `date_format`: The date format used both for displaying dates, and parsing input dates. If this option is not specified the ISO-8601 (`%Y-%m-%d`) format is used.
- `color`: By default todoman will disable colored output if stdout is not a TTY (value `auto`). Set to `never` to disable colored output entirely, or `always` to enable it regardless. This can be overridden with the `--color` option.
- `default_list`: The default list for adding a todo. If you do not specify this option, you must use the `--list` / `-l` option every time you add a todo.
- `default_due`: The default difference (in hours) between new todo's due date and creation date. If not specified, the value is 24. If set to 0, the due date for new todos will not be set.

4.2.2 Sample configuration

The below example should serve as a reference. It will read ics files from any directory inside `~/.local/share/calendars/`, and use the ISO-8601 date format (note that this is the default format, so this particular declaration is redundant).

```
[main]
# A glob expression which matches all directories relevant.
path = ~/.local/share/calendars/*
date_format = %Y-%m-%d
default_list = Personal
default_due = 1
```

4.2.3 Color and displayname

- You can set a color for each task list by creating a `color` file containing a colorcode in the format `#RRGGBB`.
- A file named `displayname` decides how the task list should be named. The default is the directory name.

See also this discussion about metadata for collections in vdirsyncer.

4.3 Usage

Todoman usage is CLI based (thought there are some TUI bits, and the intentions is to also provide a fully TUI-based interface).

First of all, the classic usage output:

```
$ todo --help
Usage: todo [OPTIONS] COMMAND [ARGS]...

Options:
  --human-time / --no-human-time  Accept informal descriptions such as
                                  "tomorrow" instead of a properly formatted
                                  date.
  --colour, --color TEXT          By default todoman will disable colored
                                  output if stdout is not a TTY (value
                                  `auto`). Set to `never` to disable colored
                                  output entirely, or `always` to enable it
                                  regardless.
  --porcelain                     Use a JSON format that will remain stable
                                  regardless of configuration or version.
  --version                       Show the version and exit.
  --help                          Show this message and exit.

Commands:
  copy    Copy tasks to another list.
  delete  Delete tasks.
  done    Mark a task as done.
  edit    Edit the task with id ID.
  flush   Delete done tasks.
  list    List unfinished tasks.
  move    Move tasks to another list.
  new     Create a new task with SUMMARY.
  show    Show details about a task.
```

The default action is `list`, which outputs all tasks for all calendars, each with a semi-permanent unique id:

```
1 [ ] ! 2015-04-30 Close bank account (0%)
2 [ ] !            Send minipimer back for warranty replacement (0%)
3 [X]   2015-03-29 Buy soy milk (100%)
4 [ ]              Fix the iPad's screen (0%)
5 [ ]              Fix the Touchad battery (0%)
```

The columns, in order, are:

- An id.
- Whether the task has been completed or not.
- An `!` indicating it's an urgent task.
- The due date
- The task summary
- The completed percentage

The id is retained by `todoman` until the next time you run the `flush` command.

To operate on a todo, the id is what's used to reference it. For example, to edit the *Buy soy milk* task from the example above, the proper command is `todo edit 3`, or `todo undo 3` to un-mark the task as done.

Editing tasks can only be done via the TUI interface for now, and cannot be done via the command line yet.

4.3.1 Synchronization

If you want to synchronize your tasks, you'll needs something that syncs via CalDAV. vdirsyncer is the recommended tool for this.

4.3.2 Interactive shell

If you install click-repl, todoman gets a new command called `repl`, which lauches an interactive shell with tab-completion.

4.3.3 Integrations

When attempting to integrate `todoman` into other systems or parse its output, you're advised to use the `--porcelain` flag, which will print all output in a pre-defined format that will remain stable regardless of user configuration or version.

The format is JSON, with one todo per line. Fields will always be present; if a todo does not have a value for a given field, it will be printed as `null`.

Fields MAY be added in future, but will never be removed.

4.4 Contributing

Bug reports and code and documentation patches are greatly appreciated. You can also help by using the development version of todoman and reporting any bugs you might encounter here.

All participants must follow the pimutils Code of Conduct.

Before working on a new feature or a bug, please browse existing issues to see whether it has been previously discussed. If the change in question is a bigger one, it's always good to open a new issue to discuss it before your starting working on it.

4.4.1 Hacking

Runtime dependencies are listed in `requirements.txt`. I recommend that you use virtualenv to make sure that no additional dependencies are required without them being properly documented.

We strictly follow the Style Guide for Python Code, which I strongly recommend you read, though you may simply run `flake8` to verify that your code is compliant.

Commits should follow Git Commit Guidelines whenever possible, including rewriting branch histories to remove any noise, and using a 50-message imperative present tense for commit summary messages.

All commits should pass all tests to facilitate bisecting in future.

4.4.2 Patch review checklist

Please follow this checklist when submitting new PRs (or reviewing PRs by others):

1. Do all tests pass?
2. Does the documentation build?
3. Does the coding style conform to our guidelines? Are there any flake8 errors?
4. Are user-facing changes documented?
5. Is there an entry for new features or dependencies in `docs/changelog.rst`?
6. Are you the patch author? Are you listed in `AUTHORS.rst`?

Hint: To quickly verify the first three items run `tox`.

Authorship

While authors must add themselves to `AUTHORS.rst`, all copyright is retained by them. Contributions are accepted under the ISC licence.

4.5 Changelog

This file contains a brief summary of new features and dependency changes or releases, in reverse chronological order.

4.5.1 v2.0.2

- Fix a crash after editing or completing a todo.

4.5.2 v2.0.1

- Fix a packaging error.

4.5.3 v2.0.0

New features

- New flag `--porcelain` for programmatic integrations to use. See the `integrations` section here for details.
- Implement a new configuration option: `default_due`.
- The configuration file is now pre-emptively validated. Users will be warned of any inconsistencies.
- The `list` command has a new `--due` flag to filter tasks due soon.
- Todo ids are now persisted in a cache. They can be manually purged using `flush`.

Packaging changes

- New runtime dependency: configobj
- New runtime dependency: python-dateutil
- New test dependency: flake8-import-order.

4.6 Licence

Todoman is licensed under the MIT licence:

```
Copyright (c) 2015-2016, Hugo Osvaldo Barrera <hugo@barrera.io>

Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
```

```
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR
OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search