hreports Documentation

Release 0.2.1

Michael Martinides

Dec 23, 2018

Contents

1 hreports **3**
- 1.1 Features . . . 3
- 1.2 Quickstart . . . 3
- 1.3 Roadmap . . . 5
- 1.4 Credits . . . 5

2 Installation **7**
- 2.1 Stable release . . . 7
- 2.2 From sources . . . 7

3 Usage **9**

4 Contributing **11**
- 4.1 Types of Contributions . . . 11
- 4.2 Get Started! . . . 12
- 4.3 Pull Request Guidelines . . . 13
- 4.4 Release Checklist . . . 13
- 4.5 Tips . . . 14

5 Credits **15**
- 5.1 Development Lead . . . 15
- 5.2 Contributors . . . 15

6 History **17**
- 6.1 0.2.1 (2018-12-23) . . . 17
- 6.2 0.2.0 (2018-04-01) . . . 17
- 6.3 0.1.5 (2018-03-05) . . . 17
- 6.4 0.1.4 (2017-12-15) . . . 17
- 6.5 0.1.3 (2017-12-15) . . . 18
- 6.6 0.1.2 (2017-12-03) . . . 18

7 Indices and tables **19**

Contents:

CHAPTER 1

hreports

A simple wrapper to create and manage reports based on hledger queries.

- Free software: MIT license
- Documentation: https://hreports.readthedocs.io.

1.1 Features

hreports saves shortcuts to hledger queries to conveniently manage multiple queries with different settings and ledger files. In addition, hreports can save the query output to pdfs with jinja templates using Pandoc.

- Conveniently create and manage multiple heldger queries
- Customize the representation of hledger query results with templates
- Save report results in pdf format
- Parametrize queries
- Use case: generate invoices with a single command, e.g. `hreports save invoice_client1`
- Use case: Manage tax reports, e.g. `hreports show tax_2017`

1.2 Quickstart

Ensure that hledger is installed. If you want to save generate pdf reports pandoc (>=1.16) and a PDF engine such as wkhtml2pdf need to be installed as well. Ensure to use the patched version wkthmltopdf to ensure https://github.com/wkhtmltopdf/wkhtmltopdf/issues/2037

Use pip to install hreports:

```
$ pip install hreports
```

The starting point of running hledger queries is having a ledger to run queries against. Imagine following simple hlegder file:

```
$ cat cash-account.ledger
1917/12/14 * Income
    assets:cash  10 USD
    income:client1

1917/12/12 * Expense
    assets:cash  5 USD
    expense:milk
```

and a hledger timesheet:

```
$ cat timesheet.ledger
1917/12/14 * Time
    (consulting:clien1)  1
```

The following command creates a report named *balance* that executes the query *hledger bal –depth 1* on the ledger *cash.ledger*:

```
$ hreports create balance --query "bal --depth 1" --ledger cash.ledger
```

When executed, hreports stores the query data in a config file for future reference. Now hreports can render the query by running:

```
$ hreports show balance
               5 USD  assets
--------------------
               5 USD
```

This makes it easy to store many different queries on different ledger files and executing them by refering to their hreports name.

1.2.1 Templating

Sometimes, it is helpful to add context to query results. hreports uses jinja templates to customize the representation of reports. The query results are added to the *output* variabel in the context of the template. In addition, the report configuration data, global configuration and custom variables are added.

Imagine the folowing simple template:

```
$ cat balance.template
The balance on {{ now }} the balance was

{{ output|last }}
```

hreports can now use this template to embed query results:

```
$ hreports show balance --template balance.template
The balance on 2017-12-15 15:39:11.519658 was:

                   5 USD
```

If you need this information for future reference, create a pdf of it by executing:

```
$ hreports save balance --template balance.template
Saved balance.pdf
```

If you keep reusing this command, simply your life by updating the report:

```
$ hreports update balance --template balance.template
```

Now hreports will always use the *balance.template* when rendering the balance report.

Templating is also helpful when you use hledger for timetracking and invoicing. Create a hreport and a simple demo template. Add a custom variable name ("hourly_rate") and value ("20") with the *-var* option:

```
$ hreports create timesheet --ledger timesheet.ledger -q "bal" -var hourly_rate 20 --
↪template invoice.template

$ cat invoice.template
{% set hours = output|last|float|round(2) %}
{% set net = hours *  hourly_rate|float  %}
Please pay me {{ net }} USD.
Signed {{ now|datetime("%Y/%m/%d") }}

$ hreports show timesheet
Please pay me 20.0 USD.
Signed on 2017/12/15
```

Admittedly, this is a somewhat simple example. But feel free to check out *heldger edit –template invoice_de.template* for a fully fledged template of a German invoice.

Finally, all report configuration data is stored in a simple YAML file which can be manipulated manually if preferred. To inspect and manipulated the config file run:

```
$ hreports edit
```

1.3 Roadmap

- Add documentation
- Add tests

1.4 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install hreports, run this command in your terminal:

```
$ pip install hreports
```

This is the preferred method to install hreports, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for hreports can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/msmart/hreports
```

Or download the tarball:

```
$ curl  -OL https://github.com/msmart/hreports/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use hreports in a project:

```
import hreports
```

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/msmart/hreports/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" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Improvements:

- Add templates
- Add version control to config files (e.g. –backup feature)

- Add git commit id of ledger file dir to global variables
- Test if requirements (hledger and pandoc) are met before running queries
- Improve error handling (e.g. template error, hledger query error, etc.) * Template Error * Hledger Query Error * YAML config file error

4.1.4 Write Documentation

hreports could always use more documentation, whether as part of the official hreports 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/msmart/hreports/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 *hreports* for local development.

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

```
$ git clone git@github.com:your_name_here/hreports.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 hreports
$ cd hreports/
$ 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 hreports tests
$ python setup.py test or 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 2.7, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/msmart/hreports/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Release Checklist

- [ ] Update HISTORY.rst
- [ ] Commit the changes:

```
git add HISTORY.rst
git commit -m "Changelog for upcoming release 0.1.1."
```

- [ ] Update version number (can also be minor or major):

```
bumpversion patch
```

- [ ] Install the package again for local development, but with the new version number:

```
python setup.py develop
```

- [ ] Run the tests:

```
tox
```

- [ ] Release on PyPI by uploading both sdist and wheel:

```
python setup.py sdist bdist_wheel
twine upload dist/*
```

- [ ] Test that it pip installs:

```
mktmpenv
pip install my_project
<try out my_project>
deactivate
```

- [ ] Push: *git push*
- [ ] Push tags: *git push –tags*

- [ ] Check the PyPI listing page to make sure that the README, release notes, and roadmap display properly. If not, copy and paste the RestructuredText into http://rst.ninjs.org/ to find out what broke the formatting.
- [ ] Edit the release on GitHub (e.g. https://github.com/msmart/hreports/releases). Paste the release notes into the release's release page, and come up with a title for the release.

4.5 Tips

To run a subset of tests:

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

CHAPTER 5

Credits

5.1 Development Lead

- Michael Martinides <msmart@posteo.de>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.2.1 (2018-12-23)

- Adds string decoding to output subprocess call
- Updates dependencies and minor fixes

6.2 0.2.0 (2018-04-01)

- Adds tests
- Improves exception handling

6.3 0.1.5 (2018-03-05)

- Adds template filter to create relative dates
- Adds multiply_last_column template filter to add an extra column which multiplies the last string on each line with a custom factor
- Adds percentage_column template filter to add an extra column which shows the percentage of the last string in a line in comparision to the buttom right value of the output
- Preserves whitespace in double quotes for check_output command
- Shows report listing in alphabetic order

6.4 0.1.4 (2017-12-15)

- Adds first template (german invoice)

- Adds exception catching for template rendering and hledger commands

6.5 0.1.3 (2017-12-15)

- Adds templating and some initial documenation.

6.6 0.1.2 (2017-12-03)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search