Tebless Documentation

Release 1.2.1

Michel Betancourt

Jun 09, 2021

Contents

Contents:

CHAPTER 1

Tebless

This library is a collection of widgets that supported from blessed allows to create interfaces in a simple way based on events.

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

1.1 Table of contents

1.2 How to install

```
pip install tebless
```

1.3 Example of usage

This will render a label containing the text 'Hello world!', centered horizontally and vertically.

```
from tebless.widgets import Label, Window, Input

@Window.decorator(main=True, min_y=10)
def view(window):
    WIDTH, HEIGHT = window.size
    def callback(evt):
        evt.store.label.value = evt.value

    window += Label(
        cordy=HEIGHT / 2 - 1,
        text='Hello world!',
        width=WIDTH,
        align='center',
        ref='label'
    )
    window += Input(
        width=WIDTH,
        cordx=WIDTH / 3,
        on_enter=callback
    )
```

so easy is placed a text in the console that changes with the input and limit min height window you can also avoid access to the store

```
from tebless.widgets import Label, Window, Input

@Window.decorator(min_y=10)
def view(window):
    WIDTH, HEIGHT = window.size

    label = Label(
        cordy=HEIGHT / 2 - 1,
        text='Hello world!',
        width=WIDTH,
        align='center'
    )

    def callback(evt):
        label.value = evt.value

    window += label
    window += Input(
        width=WIDTH,
        cordx=WIDTH / 3,
        on_enter=callback
    )
if __name__ == "__main__":
    view()
```

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 Tebless, run this command in your terminal:

```
$ pip install tebless
```

This is the preferred method to install Tebless, 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 Tebless can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/akhail/tebless
```

Or download the tarball:

```
$ curl  -OL https://github.com/akhail/tebless/tarball/master
```

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

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

CHAPTER 3

Tutorial

In this tutorial, we'll assume that tebless is already installed on your system. If that's not the case, see *Installation* .

This tutorial will walk you through these tasks:

3.1 Basic application

1. The Window widget component.
2. Creating a new basic hello world application with label widget.

3.1.1 Window widget

Before beginning we must understand how the library should work. This is separated by windows which have elements inside which draws the main component is the Window widget

This has 2 ways to use it you can use the one that seems more comfortable.

The first way using the reserved word `with`:

```python
from tebless.widgets import Window

with Window() as win:
    pass
```

As you can see it is necessary to store the window instance in a variable, this is necessary to use it in the future to add components.

The second way using the window decorator:

```python
from tebless.widgets import Window

@Window.decorator
```

(continues on next page)

(continued from previous page)

```
def main_window(win):
    pass

main_window()
```

But if you execute this right now you will get an exception `Not widgets found`. In the next section we can see how we use the label component next to window to show a hello world

3.1.2 Label widget

In the previous section we saw the window component now we can combine it with this to make our hello world. Our first step will be to create our window.:

```
from tebless.widgets import Window

@Window.decorator(main=True)
def main_window(win):
    pass
```

So far we have not seen anything new only the main property that allows us to automatically execute a window in the whole application there should only be a window with this property if you decide to use it.:

```
from tebless.widgets import Window, Label

@Window.decorator(main=True)
def main_window(window):
    window += Label(
        text='Hello world!',
    )
```

This works! It is our first hello world. But it is not the best hello world that has been seen let's make it more beautiful.:

```
from tebless.widgets import Window, Label

@Window.decorator(main=True)
def main_window(window):
    WIDTH, HEIGHT = window.size

    window += Label(
        cordy=HEIGHT / 2 - 1,
        text='Hello world!',
        width=WIDTH,
        align='center',
    )
```

Better! now we have a Label centered vertically and horizontally, window provide a property named `size` that contain the `HEIGHT` and `WIDTH` of terminal window, this is too easy!. In the next section we can see new widgets and more advance examples.

3.2 Dynamic application

1. See the input widget and properties.
2. See the menu widget.

3. Create with an input and a menu widget we obtain a filtered menu.
4. Use included filter menu.
5. Shared data between widgets with store.

3.2.1 Input widget

We have an exciting widget that allows us to capture user data is called Input let's see how it is used.:

```python
from tebless.widgets import Window, Input

@Window.decorator
def main_window(win):
    win += Input(
        label='Your name: '
    )


if __name__ == '__main__':
    main_window()
```

However, the previous example does not do anything.:

```python
from tebless.widgets import Window, Input, Label

@Window.decorator(main=True)
def view_input(window):
    label = Label(text='')

    def change_label(obj):
        label.value = f'Hello {obj.value}'

    window += Input(
        cordy=2,
        label='Your name: ',
        on_enter=change_label,
        max_len=15
    )
    window += label
```

Cool! if you enter your name, it greets you. In this example we saw a couple of new things we did not add directly to window the label in this way we can use it in the listenner function. By default the input supports 6 characters so we had to increase the size for longer names.

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/akhail/tebless/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.

4.1.4 Write Documentation

Tebless could always use more documentation, whether as part of the official Tebless 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/akhail/tebless/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 *tebless* for local development.

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

```
$ git clone git@github.com:your_name_here/tebless.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 tebless
$ cd tebless/
$ 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 tebless 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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/akhail/tebless/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_tebless
```

CHAPTER 5

Credits

5.1 Development Lead

- Michel Betancourt <MichelBetancourt23@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 1.2.0 (2017-11-24)

- Restructured project.