layabout

Release 1.0.2

Reilly Tucker Siemens

Jan 12, 2020

CONTENTS

Release v1.0.2. (*Changelog*)

> **Warning:** Layabout is deprecated. There will be no further support.

Layabout is a small event handling library on top of the Slack RTM API.

```
from pprint import pprint
from layabout import Layabout

app = Layabout()


@app.handle('*')
def debug(slack, event):
    """ Pretty print every event seen by the app. """
    pprint(event)


@app.handle('message')
def echo(slack, event):
    """ Echo all messages seen by the app except our own. """
    if event.get('subtype') != 'bot_message':
        slack.rtm_send_message(event['channel'], event['text'])


def someone_leaves(events):
    """ Return False if a member leaves, otherwise True. """
    return not any(e.get('type') == 'member_left_channel'
                   for e in events)


if __name__ == '__main__':
    # Automatically load app token from $LAYABOUT_TOKEN and run!
    app.run(until=someone_leaves)
    print("Looks like someone left a channel!")
```

CHAPTER

ONE

INSTALLATION

To install **Layabout** use pip and PyPI:

```
pip install layabout
```

CHAPTER

TWO

WHAT'S IT GOOD FOR?

You can think of **Layabout** as a micro framework for building Slack bots. Since it wraps Slack's RTM API it does best with tasks like interacting with users, responding to channel messages, and monitoring events. If you want more ideas on what you can do with it check out the examples.

CHAPTER

THREE

FEATURES

Not sold yet? Here's a list of features to sweeten the deal.

- Automatically load Slack API tokens from environment variables, provide them directly, or even bring your own `SlackClient`.
- Register multiple event handlers for one event.
- Register a single handler for multiple events by stacking decorators.
- Configurable application shutdown.
- Configurable retry logic in the event of lost connections.
- Lightweight. Depends only on the official Python slackclient library.

CHAPTER

FOUR

API REFERENCE

4.1 API Reference

4.1.1 Layabout

class layabout.**Layabout**
An event handler on top of the Slack RTM API.

Example

```
from layabout import Layabout

app = Layabout()


@app.handle('message')
def echo(slack, event):
    """ Echo all messages seen by the app. """
    channel = event['channel']
    message = event['text']
    subtype = event.get('subtype')

    # Avoid an infinite loop of echoing our own messages.
    if subtype != 'bot_message':
        slack.rtm_send_message(channel, message)

app.run()
```

handle(*type*, **, kwargs=None*)
Register an event handler with the *Layabout* instance.

Parameters

- **type** (str) – The name of a Slack RTM API event to be handled. As a special case, although it is not a proper RTM event, * may be provided to handle all events. For more information about available events see the Slack RTM API.
- **kwargs** (Optional[dict]) – Optional arbitrary keyword arguments passed to the event handler when the event is triggered.

Return type Callable

Returns A decorator that validates and registers a Layabout event handler.

Raises **TypeError** – If the decorated `Callable`'s signature does not accept at least 2 parameters.

run(*, *connector=None*, *interval=0.5*, *retries=16*, *backoff=None*, *until=None*)
Connect to the Slack API and run the event handler loop.

Parameters

- **connector** (Union[*EnvVar*, *Token*, SlackClient, None]) – A means of connecting to the Slack API. This can be an API *Token*, an *EnvVar* from which a token can be retrieved, or an established `SlackClient` instance. If absent an attempt will be made to use the `LAYABOUT_TOKEN` environment variable.
- **interval** (float) – The number of seconds to wait between fetching events from the Slack API.
- **retries** (int) – The number of retry attempts to make if a connection to Slack is not established or is lost.
- **backoff** (Optional[Callable[[int], float]]) – The strategy used to determine how long to wait between retries. Must take as input the number of the current retry and output a `float`. The retry count begins at 1 and continues up to `retries`. If absent a truncated exponential backoff strategy will be used.
- **until** (Optional[Callable[[List[dict]], bool]]) – The condition used to evaluate whether this method terminates. Must take as input a `list` of `dict` representing Slack RTM API events and return a `bool`. If absent this method will run forever.

Raises

- **TypeError** – If an unsupported connector is given.
- ***MissingToken*** – If no API token is available.
- ***FailedConnection*** – If connecting to the Slack API fails.

Return type None

4.1.2 Connectors

In addition to a `slackclient.SlackClient` these classes may be passed as a `connector` to *Layabout.run()*.

class layabout.**EnvVar**
Bases: str

A subclass for differentiating env var names from strings.

class layabout.**Token**
Bases: str

A subclass for differentiating Slack API tokens from strings.

4.1.3 Exceptions

exception layabout.**LayaboutError**
Bases: Exception

Base error for all Layabout exceptions.

exception layabout.**MissingToken**
Bases: *layabout.LayaboutError*

Raised if a Slack API token could not be found.

exception layabout.**FailedConnection**
Bases: *layabout.LayaboutError*, ConnectionError

Raised if the Slack client could not connect to the Slack API.

Inherits from both *LayaboutError* and the built-in ConnectionError for convenience in exception handling.

CHAPTER

FIVE

PROJECT INFO

5.1 Deprecation

Layabout is deprecated as of **January 1st, 2020**.

5.1.1 What?

There will be no further support for Layabout, **even for security issues**. Layabout will likely continue to work as long as as the `1.x` family of the slackclient library functions, but it is **strongly** recommended that you migrate to the `2.x` family of slackclient.

5.1.2 Why?

Please read the associated blog post that details the rationale for deprecation. **TL;DR**: The `2.x` family of the official slackclient library includes updates that make Layabout obsolete.

5.2 Why Layabout?

Why Layabout when the Slack Events API exists and there's already an officially supported events library?

Simply put, if you don't want to run a web server so Slack can call you when events happen, then Layabout is for you.

Most events are supported by the Slack RTM API as well and Layabout enables you to start handling events quickly without worrying about setting up Flask, configuring a reverse proxy, acquiring an SSL certificate, and the myriad other tasks that come with hosting a web app.

That said, if you can't afford to have a persistent WebSocket connection to the Slack API, then you probably *do* want to use the official events library.

5.3 Changelog

All notable changes to this project will be documented in this file. This change log follows the conventions of keepachangelog.com.

5.3.1 Unreleased

5.3.2 v1.0.2 (2020-01-11)

Added

- Mention of Python 3.8 functionality.
- Information about how to acquire Slack API tokens in examples.

Deprecated

- Layabout is deprecated. There will be no further support.

5.3.3 v1.0.1 (2018-10-14)

Added

- Official support for Python 3.7.

Changed

- Improve the examples to warn users when an API token is missing.
- Improve the format of `CHANGELOG.rst`.

5.3.4 v1.0.0 (2018-06-18)

Initial release.

Contributors

- Geoff Shannon (@RadicalZephyr)
- Mike Canoy (@solus-impar)
- Sophie Anderson (@SophieKAn)
- Tucker Siemens (@reillysiemens)

5.4 License

5.5 Authors

5.5.1 Lead Developers

- Tucker Siemens (@reillysiemens)

5.5.2 Contributors (alphabetical)

These folks have lent their support to the project in some way.

- Alex LordThorsen (@rawrgulmuffins)
- Geoff Shannon (@RadicalZephyr)
- Kyle Rader (@kyle-rader)
- Mike Canoy (@solus-impar)
- Sophie Anderson (@SophieKAn)

5.6 Contributing Guidelines

Please follow these guidelines for contributing to this project.

5.6.1 Repository Management

- Fork this project into your own repository.
- Follow the *version control* guidelines.
- No changes should reach the `master` branch except by way of a pull request.

Submitting Issues

Create an issue to report bugs or request enhancements. Better yet, fix the problem yourself and submit a pull request.

Bug Reports

When reporting a bug, please provide the steps to reproduce the problem and any details that could be important such as whether this is the first time this has happened or whether others are experiencing it.

Pull Requests

PRs must remain focused on fixing or addressing one thing (see topic branch model). Make sure your pull request contains a clear title and description. Test coverage should not drop as a result. If you add code, you add tests.

Be sure to follow the guidelines on *writing code* if you want your work considered for inclusion.

Handling Pull Requests

- Pull Requests **must** include:
  - Title describing the change
  - Description explaining the change in detail
  - Tests
- A maintainer will respond to Pull Requests with one of:
  - 'Ship It', 'LGTM', , or some other affirmation
  - What must be changed
  - Won't accept and why

Nota Bene

- PRs which have titles beginning with `[WIP]` (indicating a Work In Progress) status will **not** be merged until the `[WIP]` prefix has been removed from the title. This is a good way to get feedback from maintainers if you are unsure of the changes you are making.
- A PR that has received a 'Ship It' may not be merged immediately.
- You may be asked to rebase or squash your commits to keep an orderly version control history.

5.6.2 Writing Code

Writing code is a creative process and there will always be exceptions to the rules, but it's good to maintain certain standards. In general, please follow these code conventions.

Coding Style

- Follow **PEP 8** guidelines.
- Code in this project **must** be linted. For convenience the project Makefile contains a `lint` target to run the linting process for you.
- Try to respect the style of existing code.

Version Control

- Fork the central repository and work from a clone of your own fork.
- Follow the topic branch model and submit pull requests from branches named according to their purpose.
- Review the GitHub Flow documentation and, in general, try to stick to the principles outlined there.

Testing

- Code **must** be tested. Write or update related unit tests so you don't have to manually retest the same thing many times.
- Tests for this project are written using the pytest framework. While it isn't always achievable this project strives to maintain 100% test coverage. For convenience the project Makefile contains a `test` target to run the tests and generate coverage for you.
- In addition to unit testing code in this project is statically type checked using mypy. For convenience the project Makefile contains a `type-check` target to run mypy for you.

Documentation

- Public interfaces **must** be thoroughly documented. At a minimum this includes inputs, return types, exceptions raised, and surprising behavior like state changes.
- Documentation for this project is written in reStructuredText and generated with Sphinx. For convenience the project Makefile contains a `docs` to run Sphinx for you.

5.7 Code of Conduct

5.7.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

5.7.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

5.7.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

5.7.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

5.7.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team via email. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

5.7.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available on their website.

Note: Layabout **only** supports Python 3.6+ and will never be backported to Python 2. If you haven't moved over to Python 3 yet please consider the many reasons to do so.

PYTHON MODULE INDEX

INDEX