Friday Documentation

Release 0.3.5

Isaac Luke Smith

Jul 03, 2021

Contents

Contents:

CHAPTER 1

Friday

[![Run on Repl.it](https://repl.it/badge/github/Zenohm/Friday)](https://repl.it/github/Zenohm/Friday) .. image:: https://img.shields.io/travis/Zenohm/Friday.svg

target https://travis-ci.org/Zenohm/Friday

An open source virtual assistant. Uses the https://api.ai/ system for backend analysis of requests.

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

1.1 Features

- Easy to use and flexible plugin system.
- Google's speech to text and text to speech system.
- Support for Windows, Linux, and OS/X.
- Python 2 and 3 support.
- Fully automated tests using the wonderful Travis CI build system.
- A simple codebase.

1.2 Credits

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

CHAPTER 2

Installation

2.1 Stable release

To install Friday, run this command in your terminal:

```
$ pip install https://github.com/Zenohm/Friday/archive/master.zip
```

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

You can either clone the public repository:

```
$ git clone git://github.com/Zenohm/friday
```

Or download the tarball:

```
$ curl  -OL https://github.com/Zenohm/friday/tarball/master
```

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

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

CHAPTER 3

Usage

To use Friday in a project:

```
from friday import friday
bot = friday.Friday()
```

From here, the bot can be interacted with directly in order to test functionality.

To run Friday from the command line:

```
# Navigate to the Friday/ directory.
python -m friday.cli
```

CHAPTER 4

Plugins

4.1 Plugin format

Every plugin has to have a certain layout in order to function properly. In future, the plugin system might be modified slightly to take advantage of object-oriented design to reduce the complexity of individual plugin classes.

4.1.1 File layout

For now, each plugin must follow the default layout for a yapsy plugin:

For a Standard plugin:

- **myplugin.yapsy-plugin**
  - A plugin info file identical to the one previously described.
- **myplugin**
  - A directory containing an actual Python plugin (ie with a __init__.py file that makes it importable). The upper namespace of the plugin should present a class inheriting the IPlugin interface (the same remarks apply here as in the previous case).

For a Single file plugin:

- **myplugin.yapsy-plugin**
  - A plugin info file which is identified thanks to its extension, see the Plugin Info File Format to see what should be in this file. The extension is customisable at the PluginManager's instanciation, since one may usually prefer the extension to bear the application name.
- **myplugin.py**
  - The source of the plugin. This file should at least define a class inheriting the IPlugin interface. This class will be instanciated at plugin loading

4.1.2 Class layout

The class which inherits from the IPlugin interface, as specified above, should have the following abstract setup:

```
from yapsy.IPlugin import IPlugin


class Plugin(IPlugin):
    def can_perform(self, friday, request) -> bool:
        pass

    def perform(self, friday, request):
        pass
```

4.1.3 Method descriptions

There are two methods which every plugin must have:

can_perform Determines whether the input plugin should be executed based on the current input.

perform Executes the functionality of the plugin

These two methods both receive the same input:

self A reference to the plugin's instance

friday A reference to the instance of the assistant, allowing plugins to only be executed when the assistant is in a specific state

request The response from the API.AI service formatted as a Python dictionary.

Using these inputs, each plugin has access to everything the assistant has access to, and full-featured plugins can be built using this system.

For more references, look at some example plugins included with this project.

CHAPTER 5

Future plans

5.1 Plugins

- Input plugins which allow different front-ends to be used to communicate with the assistant.
- Loading plugins which allow the assistant to delay the user while the back-end is processing.
- Output plugins which allow the assistant to provide responses using different output systems like a GUI or text to speech.

5.2 Functionality

- Maybe allow different back-ends to be used (Eg. *wit.ai*)
- Fully functioning tests with example api.ai output
- More local functionality, allowing the system to do more without an internet connection
- Backup systems that work without internet
- Asynchronous behavior

CHAPTER 6

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:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/Zenohm/friday/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.

6.1.2 Fix Bugs

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

6.1.3 Implement Features

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

6.1.4 Write Documentation

Friday could always use more documentation, whether as part of the official Friday docs, in docstrings, or even on the web in blog posts, articles, and such.

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/Zenohm/friday/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 :)

6.2 Get Started!

Ready to contribute? Here's how to set up *friday* for local development.

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

```
$ git clone git@github.com:your_name_here/friday.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 friday
$ cd friday/
$ 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 friday 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.

6.3 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, 3.5, and 3.6. Check https://travis-ci.org/Zenohm/friday/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ py.test tests.test_friday
```

CHAPTER 7

Credits

7.1 Development Lead

- I. Smith <senzenpen@gmail.com>

7.2 Contributors

None yet. Why not be the first?

CHAPTER 8

History

8.1 0.3.3 (2017-01-12)

- First integration of Travis CI and code review tools

8.2 0.3.4 (2017-01-15)

- All Travis CI checks properly integrated
- Limited use from the command line available
- More useful documentation

CHAPTER 9

Indices and tables

- genindex
- modindex
- search