phrasebook Documentation

Release 0.0.3

patdaburu

Mar 29, 2019

Contents:

Store phrases (SQL, messages, what-have-you) alongside your modules.

CHAPTER 1

Getting Started

1.1 Installing the Library

You can use pip to install *phrasebook*.

```
pip install phrasebook
```

1.2 Now What?

Check out *the tutorial*.

CHAPTER 2

Tutorial

phrasebook is designed to make it relatively simple for you to place files containing string templates alongside your python modules. You have a few different options for doing so: You can place all your strings in a *single file* or you can break them out into *multiple files within a directory*. While we've tried to use simple naming conventions to minimize the boiler plate for creating a *single phrasebook for a module*, you can also create as many phrasebooks as you need by *specifying the path* for each one.

2.1 Single File Phrasebook

If you have just a few simple strings, you can place them all in a single toml file.

Listing 1: my_module.phr

```
txt1 = "Hello, $NAME."
txt2 = "Hello, $YOURNAME.  My name is $MYNAME."

[sub1]
txt1 = "Hello again, $NAME"
txt2 = "Hello again, $YOURNAME.  My name is still $MYNAME."
```

You can then access it from a python module with the same base name.

Note: If you don't supply the *path* constructor argument, the `Phrasebook` class will attempt to find a file or directory with the same name as the current module with the *.phr* suffix.

Listing 2: my_module.py

```
from phrasebook import Phrasebook

phrasebook = Phrasebook().load()
```

(continues on next page)

(continued from previous page)

```
print(phrasebook.substitute('txt1', NAME='Eric'))
print(phrasebook.substitute('txt2', YOURNAME='Eric', MYNAME='Terry'))

print(phrasebook.substitute('sub1.txt1', NAME='Eric'))
print(phrasebook.substitute('sub1.txt2', YOURNAME='Eric', MYNAME='Terry'))
```

Listing 3: output

```
Hello, Eric.
Hello, Eric.  My name is Terry.
Hello again, Eric
Hello again, Eric.  My name is still Terry.
```

2.2 Phrasebook Directory

Listing 4: my_module.phr/query1.sql

```
SELECT $COLUMN FROM $TABLE
```

Listing 5: my_module.phr/sub1/query2.sql

```
SELECT $COLUMN1, $COLUMN2 FROM $TABLE
```

Listing 6: my_module.py

```
from phrasebook import Phrasebook

phrasebook = Phrasebook().load()

print(phrasebook.substitute('query1', COLUMN='first', TABLE='names'))
print(
    phrasebook.substitute(
        'sub1.query2',
        COLUMN1='first',
        COLUMN2='last',
        TABLE='names'
    )
)
```

Listing 7: output

```
SELECT first FROM names
SELECT first, last FROM names
```

2.3 Specifying the Path

You may not always want your phrasebooks to reside alongside your modules; sometimes you may want to share phrasebooks across modules. In those cases, you can provide a *path* argument to indicate the file or directory that contains your phrases.

```
phrasebook = Phrasebook(path='/path/to/my/phrases.phr').load()
```

2.4 Specifying Suffixes

If you have a phrases directory that contains many different types of files, you can indicate which files you want to include by specifying their extensions using the *suffixes* constructor argument. You can use this convention if you need to put other types of files (perhaps something like a *README.md* file that provides some documentation for the phrases) alongside the phrase files.

```
phrasebook = Phrasebook(suffixes=['.sql']).load()
```

Note: The example above demonstrates how you might create a phrase book that is particular to *SQL* phrases, but there is also a built-in *SqlPhrasebook* that you can use for that particular purpose.

CHAPTER 3

Using the Command Line Application

This project contains a command line application (*phrasebook*) based on Click.

3.1 Installation

The command line application is installed automatically when the package is installed.

3.2 Running the CLI in the Development Environment

If you need to run the application from within the project's own development environment, you can use the *make build* target.

```
make build
```

3.3 Getting Help

The command line application has a help function which you can access with the *–help* flag.

```
phrasebook --help
```

CHAPTER 4

API Documentation

4.1 phrasebook

Note: Most of the functions and objects defined in the *phrasebook.phrasebook* module are actually available in the top-level (*phrasebook*) namespace.

Store phrases (SQL, messages, what-have-you) alongside your modules.

phrasebook.phrasebook.**PHRASES_SUFFIX = '.phr'**
the standard suffix for phrasebook directories

class phrasebook.phrasebook.**Phrasebook**(*path: str = None*, *suffixes: Iterable[str] = None*)
Bases: object

A phrasebook is an indexed collection of string templates.

__init__(*path: str = None*, *suffixes: Iterable[str] = None*)

Parameters

- **path** – the path to the phrases directory, or a file that has an accompanying phrasebook directory
- **suffixes** – the suffixes of phrase files

See also:

Python's String Templates

get(*phrase: str*, *default: str = None*) → string.Template
Get a phrase template.

Parameters

- **phrase** – the name of the phrase template
- **default** – a default template or string

Returns the template (or the default), or *None* if no template is defined

See also:

gets()

gets(*phrase: str*, *default: str = None*) → str
Get a phrase template string.

Parameters

- **phrase** – the name of the phrase template
- **default** – a default template or string

Returns the template (or the default), or *None* if no template is defined

items() → ItemsView[str, string.Template]
Get the key-value pairs.

load() → phrasebook.phrasebook.Phrasebook
Load the phrases.

Returns this instance

path
Get the path to the phrases directory.

substitute(*phrase: str*, *default: str = None*, *safe: bool = True*, ***kwargs*) → str
Perform substitutions on a phrase template and return the result.

Parameters

- **phrase** – the phrase
- **default** – a default template
- **safe** – *True* (the default) to leave the original placeholder in the template in place if no matching keyword is found
- **kwargs** – the substitution arguments

Returns the substitution result

suffixes
Get the recognized suffixes for phrase files.

4.2 phrasebook.sql

SQL phrases (y'know… like queries and query fragments and such…)

class phrasebook.sql.**SqlPhrasebook**(*path: str = None*, *suffixes: Iterable[str] = ('.sql',)*)
Bases: *phrasebook.phrasebook.Phrasebook*

A SQL phrasebook is an indexed collection of string templates that is particular to SQL phrases.

__init__(*path: str = None*, *suffixes: Iterable[str] = ('.sql',)*)

Parameters

- **path** – the path to the phrases directory, or a file that has an accompanying phrasebook directory
- **suffixes** – the suffixes of phrase files

See also:

Python's String Templates

CHAPTER 5

Development

5.1 Getting Started

This section provides instructions for setting up your development environment. If you follow the steps from top to bottom you should be ready to roll by the end.

5.1.1 Get the Source

The source code for the *phrasebook* project lives at github. You can use *git clone* to get it.

```
git clone https://github.com/patdaburu/phrasebook
```

5.1.2 Create the Virtual Environment

You can create a virtual environment and install the project's dependencies using *make*.

```
make venv
make install
source venv/bin/activate
```

5.1.3 Try It Out

One way to test out the environment is to run the tests. You can do this with the *make test* target.

```
make test
```

If the tests run and pass, you're ready to roll.

5.1.4 Getting Answers

Once the environment is set up, you can perform a quick build of this project documentation using the *make answers* target.

```
make answers
```

5.2 Using the *Makefile*

This project includes a `Makefile` that you can use to perform common tasks such as running tests and building documentation.

5.2.1 Targets

This section contains a brief description of the targets defined in the `Makefile`.

clean

Remove generated packages, documentation, temporary files, *etc*.

lint

Run pylint against the project files.

test

Run the unit tests.

docs

Build the documentation for production.

answers

Perform a quick build of the documentation and open it in your browser.

package

Build the package for publishing.

publish

Publish the package to your repository.

build

Install the current project locally so that you may run the command-line application.

venv

Create a virtual environment.

install

Install (or update) project dependencies.

licenses

Generate a report of the projects dependencies and respective licenses.

Note: If project dependencies change, please update this documentation.

5.3 Publishing the Package

As you make changes to the project, you'll probably want to publish new version of the package. *(That's the point, right?)*

5.3.1 Publishing

The actual process of publishing the project is just a matter of running the *publish* target.

```
make publish
```

5.3.2 Installing

If you just need to install the library in your project, have a look at the *general tutorial* article.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

CHAPTER 7

Python Module Dependencies

The `requirements.txt` file contains this project's module dependencies. You can install these dependencies using `pip`.

```
pip install -r requirements.txt
```

7.1 requirements.txt

```
click>=7.0,<8
pip-check-reqs>=2.0.1,<3
pip-licenses>=1.7.1,<2
pylint>=1.8.4,<2
pytest>=3.4.0,<4
pytest-cov>=2.5.1,<3
pytest-pythonpath>=0.7.2,<1
setuptools>=38.4.0
Sphinx==1.7.2
sphinx-rtd-theme==0.3.0
toml>=0.10.0
tox>=3.0.0,<4
twine>=1.11.0,<2
```

7.2 Runtime Dependencies and Licenses

Name	Version	License	URL
Click	7.0	BSD	https://palletsprojects.com/p/click/
bleach	3.1.0	Apache Software License	https://github.com/mozilla/bleach
filelock	3.0.10	License	https://github.com/benediktschmitt/py-filelock
readme-renderer	24.0	Apache License, Version 2.0	https://github.com/pypa/readme_renderer
toml	0.10.0	MIT	https://github.com/uiri/toml
webencodings	0.5.1	BSD	https://github.com/SimonSapin/python-webencodings

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

G

I

L

P

S