snippy-tldr

Release 0.2.0

Heikki J. Laaksonen

Apr 05, 2020

CONTENTS:

CHAPTER

ONE

INTRODUCTION

Snippy-tldr is plugin to import tldr man pages for snippy tool. This is a plugin that itself does not have any functionality without the snippy tool.

CHAPTER

TWO

INSTALLATION

To install, run:

```
pip install snippy-tldr --user

# Export user local Python package bin to PATH if needed.
export PATH=${PATH}:~/.local/bin
```

CHAPTER

THREE

DEVELOPMENT

3.1 Installation

The instructions are tested with Fedora 30 and Bash shell.

Note: The instructions install virtual environments with `python3` and add the Python 3 virtual environment modules to Python user script directory. This allows for example creating own Linux user for Snippy-tldr development which has an isolated virtual environment setup from global Python modules.

In case you want different virtual environment setup, you have to modify the examples.

The virtual environments are installed under `${HOME}/.cache/snippy-tldr`.

Note: The installation instructions add new software packages. Execute at your own risk.

3.1.1 Fedora

Follow the instructions to install the project on a Fedora Linux.

```
# Clone the project from the GitHub.
mkdir -p ${HOME}/.cache/snippy-tldr
mkdir -p ${HOME}/.local/share/snippy-tldr
mkdir -p ${HOME}/devel/snippy-tldr && cd $_
git clone https://github.com/heilaaks/snippy-tldr.git .

# Install CPython versions.
sudo dnf install -y \
    python27 \
    python34 \
    python35 \
    python36 \
    python37 \
    python38 \
    python3-devel \
    python2-devel

# Upgrade CPython versions.
sudo dnf upgrade -y \
    python27 \
```

(continues on next page)

(continued from previous page)

```
    python34 \
    python35 \
    python36 \
    python37 \
    python38 \
    python3-devel \
    python2-devel

# Install PyPy versions.
sudo dnf install -y \
    pypy2 \
    pypy3 \
    pypy2-devel \
    pypy3-devel \
    postgresql-devel

# Upgrade PyPy versions.
sudo dnf upgrade -y \
    pypy2 \
    pypy3 \
    pypy2-devel \
    pypy3-devel \
    postgresql-devel

# Below are 'generic instructions' that can be used also with other
# Linux distributions.

# Upgrade pip.
pip install --upgrade pip

# Install Python virtual environments.
pip3 install --user --upgrade \
    pipenv \
    virtualenv \
    virtualenvwrapper

# Enable virtualenvwrapper and add the Python user script directory
# to the path if needed.
vi ~/.bashrc
    # Snippy-tldr development settings.
    [[ ":$PATH:" != *"${HOME}/.local/bin"* ]] && PATH="${PATH}:${HOME}/.local/bin"
    export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
    export VIRTUALENVWRAPPER_VIRTUALENV=${HOME}/.local/bin/virtualenv
    export WORKON_HOME=${HOME}/.cache/snippy-tldr/.virtualenvs
    source virtualenvwrapper.sh
    cd ${HOME}/devel/snippy-tldr
    workon snippy-tldr-python3.7
source ~/.bashrc

# Create virtual environments.
for PYTHON in python2.7 \
              python3.4 \
              python3.5 \
              python3.6 \
              python3.7 \
              python3.8 \
              pypy \
```

(continues on next page)

(continued from previous page)

```
              pypy3
do
    if which ${PYTHON} > /dev/null 2>&1; then
        printf "create snippy-tldr venv for ${PYTHON}\033[39G: "
        mkvirtualenv --python $(which ${PYTHON}) snippy-tldr-${PYTHON} > /dev/null 2>&
↪1
        if [[ -n "${VIRTUAL_ENV}" ]]; then
            printf "\033[32mOK\033[0m\n"
        else
            printf "\e[31mNOK\033[0m\n"
        fi
        deactivate > /dev/null 2>&1
    fi
done

# Install virtual environments. Some versions are not pinned.
for VENV in $(lsvirtualenv -b | grep snippy-tldr-py)
do
    workon ${VENV}
    printf "deploy venv ${VENV}\033[39G: "
    if [[ ${VIRTUAL_ENV} == *${VENV}* ]]; then
        make upgrade-wheel PIP_CACHE=--no-cache-dir
        make install-devel PIP_CACHE=--no-cache-dir
        printf "\033[32mOK\033[0m\n"
    else
        printf "\e[31mNOK\033[0m\n"
    fi
    deactivate > /dev/null 2>&1
done

# Example how to delete Snippy-tldr virtual environments.
deactivate > /dev/null 2>&1
for VENV in $(lsvirtualenv -b | grep snippy-tldr-py)
do
    printf "delete venv ${VENV}\033[39G: "
    rmvirtualenv ${VENV} > /dev/null 2>&1
    printf "\033[32mOK\033[0m\n"
done
```

3.2 Workflows

3.2.1 Testing

For the snippy-tldr development, prefer a virtual environment with the latest Python release and Python 2.7. The continuous integration will run all the tests against all supported Python version but the most problems can be captured by testing with the latest Python 3 version and Python 2.7.

```
# Work in a Python virtual environment.
workon snippy-tldr-python3.7
```

The snippy-tldr continuous integration will run all tests the same make tests target.

```
# Run the development tests.
make tests
```

3.2.2 Documentation

The documentation includes manual and automated documentation. Automated documentation is extracted from source code docstrings.

```
# Create documents.
make docs

# Open the document in a web brower.
file:///<home>/devel/snippy-tldr/docs/build/html/development.html
```

3.3 Relasing

```
# Test PyPI installation before official release into PyPI.
make clean-all
python setup.py sdist bdist_wheel
twine check dist/*
twine upload --repository-url https://test.pypi.org/legacy/ dist/*

# Create a tag where the version follows semating versioning 2.0.0.
git tag -a 0.1.0 -m "Add new release 0.1.0"
git push -u origin 0.1.0

# Releast into PyPi.
make clean-all
python setup.py sdist bdist_wheel
twine upload dist/*
```

3.4 Modules

3.4.1 snippy_tldr.plugin

Description

Design

Snippy-tldr is a plugin to import tldr man pages for Snippy.

snippy_tldr.plugin.**snippy_import_hook**(*logger*, *infile*)

Import content for Snippy tool.

This is an import hook that must return an iterator object. The iterator must be iterable class that implements `next` and `len` methods. The JSON structures stored in the iterator must pass the `validate` method in the `snippy.plugins.Schema` class.

The `snippy.plugins.Parser` class may be used to parse the source data to a JSON content for Snippy.

```
# Tldr man pages hierarchical layers.
#
# translations       platforms          pages
# ============       =========        =========
  pages.it      |
```

(continues on next page)

(continued from previous page)

```
  pages.pt-BR  |
  pages.zh     |
  pages        +---+  common   |
                   |  linux    |
                   |  osx      |
                   |  sunos    |
                   +  windows  +---+  alpine.md
                                  |  apk.md
```

Term	Decscription
page	One tldr page like `alpine.md` or `apk.md`.
platform	One platform like `linux` or `osx`.
translation	One tldr man page translation like `pages.it` or `pages.zh`.

Parameters

- **logger** (*obj*) – Logger to be used with the plugin.
- **infile** (*str*) – Value from the Snippy `--file` command line option.

Returns Iterator object that stores the content from plugin to Snippy tool.

Return type obj

Examples

```
>>> from snippy.plugins import Const
>>> from snippy.plugins import Parser
>>> from snippy.plugins import Schema
>>>
>>> class SnippyTldr(object):
>>>
>>>     def __init__(self, logger, file):
>>>         self._logger = logger
>>>         self._uri = file
>>>         self._schema = Schema()
>>>         self._content = []
>>>         self._i = 0
>>>
>>>         self._read_tldr_files()
>>>
>>>     def __len__(self):
>>>         return len(self._content)
>>>
>>>     def __iter__(self):
```

(continues on next page)

(continued from previous page)

```
>>>         return self
>>>
>>>     def next(self):
>>>         if self._i < len(self):
>>>             content = self._content[self._i]
>>>             self._i += 1
>>>         else:
>>>             raise StopIteration
>>>
>>>         return content
>>>
>>>     __next__ = next  # Python 3 compatible iterator.
>>>
>>>     def _read_tldr_files(self):
>>>         with open('alpine.md', 'w') as infile:
>>>             tldr = self._parse_file(infile.read())
>>>             if self._schema.validate(tldr):
>>>                 self.notes.append(tldr)
>>>
>>>     def _parse_file(self, infile)
>>>         content = {}
>>>         content['category'] = Const.SNIPPET
>>>         content['data'] = Parser.format_data(['first line', 'second line'])
>>>
>>>         return content
```

class snippy_tldr.plugin.**SnippyTldr**(*logger*, *uri*)

Plugin to import tldr man pages for snippy.

next()

Return the next tldr man page.

The returned pages are pre-formatted for Snippy tool.

Returns The next tldr mage in interator.

Return type dict

_get_uri(*uri*)

Format URI from the user.

This method makes sure that the URI or path received from user is in correct format.

The trailing slash is added if the URI is not pointing to a file. The trailing slash allows `ulrjoin` to add path objects like filenames to the URI without removing the last object in the URI path.

Parameters **uri** (*str*) – URI or path received from the `--file` CLI option.

Returns Formatted URI for the plugin.

Return type str

_read_tldr_pages()

Read all `tldr pages` from the URI.

_get_tlrd_pages(*uri*)

Get all `tldr pages`.

Read all tldr pages from the given URI. The pages are returned in a dictionary that contains keys for translations and tldr platforms. The tldr pages are in a list of full GitHub raw URLs under each platform.

Parameters **uri** (*str*) – URI where the tldr pages are read.

Returns All tldr pages with GitHib raw URL.

Return type dict

_get_github_tldr_pages(*branch*, *translations*, *platforms*)
Get tldr pages.

Method takes a list of translations and platforms to try to read all the needed tldr pages with as less GitHub API requests as possible.

Parameters

- **branch** (*str*) – GitHub branch.
- **translations** (*tuple*) – List of translations to read under the branch.
- **platforms** (*tuple*) – List of platforms to read under the branch.

Returns Tldr pages under given translations and platforms.

Return type dict

_read_pages(*platform*, *pages*)
Read tldr pages from the platform.

Parameters

- **platform** (*str*) – A tldr platform where the pages are read.
- **pages** (*tuple*) – List of tldr pages under the platform.

_read_tldr_page(*uri*, *platform*)
Read a tldr page.

Parameters

- **uri** (*str*) – URI or path where the tldr file is read.
- **platform** (*str*) – Platform where the page is stored.

static _join_paths(*uri*, *path_object*)
Join URI or path to an object.

Parameters

- **uri** (*str*) – URI or path base.
- **path_object** (*str*) – Path object to be added to the URI.

Returns Joined URI or path.

Return type str

is_api_error(*http*)
Test if GitHub API response was an error.

Parameters **http** (*obj*) – Request package response object.

Returns True in case of REST API error response.

Return type bool

_parse_tldr_page(*source*, *platform*, *page*)
Parse and validate one tldr man page.

The method parses and validates one tldr man page to a snippet data structure for the Snippy tool.

Parameters

- **source** (*str*) – A link where the tldr man page was read.
- **platform** (*str*) – The platfrom where the tldr page belongs.
- **page** (*str*) – A tldr man page in a text string.

Returns Validated JSON structure from a tldr man page.

Return type dict

_read_tldr_data(*tldr*)
Parse and format tldr man page `data` attribute.

Args tldr (str): A tldr snippet in a text string.

Returns Formatted list of tldr man page snippets.

Return type tuple

_read_tldr_brief(*tldr*)
Parse and format tldr man page `brief` attribute.

Args tldr (str): A tldr snippet in a text string.

Returns Utf-8 encoded unicode string.

Return type str

_read_tldr_description(*tldr*)
Parse and format tldr man page `description` attribute.

Args tldr (str): A tldr snippet in a text string.

Returns Utf-8 encoded unicode string.

Return type str

_read_tldr_name(*tldr*)
Parse and format tldr man page `name` attribute.

Args tldr (str): A tldr snippet in a text string.

Returns Utf-8 encoded unicode string.

Return type str

static _format_list(*data*)
Remove empty strings and trim newlines from a list.

Args data (list): List of strings.

Returns Formatted list of tldr man page snippets.

Return type list

_format_brief(*brief*)
Format brief description for tldr man page.

Remove additional Markdown tokens like '>' and limit the length of the string to be more suitable for content `brief` attribute.

The last dot is removed because it is not considered part of the `brief` attribute for styling issue.

Args brief (str): Brief read from the tldr man page.

Returns Tldr specific format for the `brief` attribute.

Return type str

_format_description(*description*)
Format tldr man page description.

Remove additional Markdown tokens like '>' from the description.

Args description (str): Description read from the tldr man page.

Returns Tldr specific format for the `description` attribute.

Return type str

static _limit_string(*string_*, *len_*)
Limit the string length

PYTHON MODULE INDEX

S

INDEX

Symbols

I

N

S