spinnerchief Documentation

Release 0.1.1

April 02, 2014

Spinner Chief is an online service for spinning text (synonym substitution) that creates unique version(s) of existing text. This package provides a way to easily interact with SpinnerChief API. Usage requires an account, get one here and an api key which you get by registering a developer account.

- Source code @ GitHub

CHAPTER 1

Install within virtualenv

```
$ virtualenv foo
$ cd foo
$ git clone https://github.com/niteoweb/spinnerchief
$ bin/pip install spinnerchief/

# running tests:
$ bin/pip install unittest2 mock
$ bin/python -m unittest discover -s spinnerchief/src/spinnerchief/tests
```

CHAPTER 2

Buildout

```
$ git clone https://github.com/niteoweb/spinnerchief
$ cd spinnerchief
$ python bootstrap.py
$ bin/buildout

# running tests:
$ bin/py -m unittest discover -s src/spinnerchief/tests

# check code for imperfections
$ bin/vvv src/spinnerchief
```

CHAPTER 3

Usage

```
>>> import spinnerchief
>>> sc = spinnerchief.SpinnerChief("<yourapikey>", "<yourusername>", "<yourpassword>")

>>> sc.text_with_spintax(text="My name is Ovca!")
"{I am|I'm|My friends call me|Throughout southern california|Im} Ovca!"

>>> sc.unique_variation(text="My name is Ovca!")
"Im Ovca!"

>>> print "used: %s" % sc.quota_used()
used: 2

>>> print "left: %s" % sc.quota_left()
left: 18
```

CHAPTER 4

API

4.1 API

4.1.1 SpinnerChief

class spinnerchief.**SpinnerChief**(*apikey*, *username*, *password*)

A class representing the Spinner Chief API (http://developer.spinnerchief.com/API_Document.aspx).

Articles must be in Unicode object type.

URL = u'http://api.spinnerchief.com:9001/apikey={apikey}&username={username}&password={password}&'

URL for invoking the API

_get_param_value(*param_name*, *params*)

Returns parameter value or use default.

_send_request(*text=''*, *params={'pos': '0', 'usehurricane': '1', 'spinhtml': '0', 'protecthtml': '0', 'percent': '0', 'phrasecount': '2', 'Chartype': '1', 'replacetype': '0', 'autospin': '1', 'thesaurus': 'English', 'convertbase': '0', 'Orderly': '0', 'Wordscount': '5', 'spinfreq': '4', 'tagprotect': '[]', 'spintype': '0', 'UseGrammarAI': '0', 'protectwords': None, 'rule': 'none', 'onecharforword': '0', 'wordquality': '0', 'original': '0'}*)

Invoke Spinner Chief API with given parameters and return its response.

Parameters **params** (*dictionary*) – parameters to pass along with the request

Returns API's response (article)

Return type string

_validate(*params*)

Checks every single parameter and raise error on wrong key or value.

_value_has(*param*, *values*, *params*)

Raise WrongParameterVal if value of param is not in values.

_value_is_int(*param*, *params*)

Raise WrongParameterVal if value of param is not integer.

quota_left()

The server returns today's remaining query times of this account.

quota_used()

The server returns today's used query times of this account.

text_with_spintax(*text*, *params=None*)
Return processed spun text with spintax.

Parameters

- **text** (*string*) – original text that needs to be changed
- **params** (*dictionary*) – parameters to pass along with the request

Returns processed text in spintax format

Return type string

unique_variation(*text*, *params=None*)
Return a unique variation of the given text.

Parameters

- **text** (*string*) – original text that needs to be changed
- **params** (*dictionary*) – parameters to pass along with the request

Returns processed text

Return type string

4.1.2 Exceptions

exception spinnerchief.exceptions.**LoginError**(*api_error_msg*)
Raised if there are login errors.

exception spinnerchief.exceptions.**NetworkError**(*msg*)
Raised if there are network problems, like timeout.

exception spinnerchief.exceptions.**SpinnerChiefError**(*api_error_msg*)
Base class for exceptions in Spinner Chief module.

exception spinnerchief.exceptions.**WrongParameterName**(*name*)
Raised on unsuppported parameter name.

exception spinnerchief.exceptions.**WrongParameterVal**(*name*, *val*)
Raised on invalid parameter value.

CHAPTER 5

Developer documentation

5.1 Developer environment

5.1.1 Dependencies

A C/C++ compilation environment

On a Debian based system install the `build-essential` package. On OS X, install XCode and MacPorts.

Git

On a Debian based system install the `git-core` package. On OS X, get the latest version from http://code.google.com/p/git-osx-installer/.

Python 2.7

On a Debian based system install the `python2.7-dev` package. On OS X (and others) use the buildout.python to prepare a clean Python installation.

5.1.2 Build

First, you need to *clone* the git repository on GitHub to download the code to your local machine:

```
$ git clone git@github.com:niteoweb/spinnerchief.git
```

What follows is initializing the *buildout* environment:

```
$ cd spinnerchief
$ python2.7 bootstrap.py
```

And now you can *run the buildout*. This will fetch and configure tools and libs needed for developing *spinnerchief*:

```
$ bin/buildout
```

5.1.3 Verify

Your environment should now be ready. Test that by using the `py` Python interpreter inside the `bin` directory, which has *spinnerchief* installed in it's path:

```
$ bin/py

>>> from spinnerchief import SpinnerChief
>>> sc = SpinnerChief('api_key', 'username', 'password')
>>> sc.unique_variation('Some random text.')
u"Some random lines."
```

The code for *spinnerchief* lives in `src/`. Make a change and re-run `bin/py` to see it resembled!

Moreover, you should have the following tools in the `bin/` directory, ready for use:

vvv

A sintax validation tool.

sphinbuilder

A tool for testing HTML render of *spinnerchief*'s documentation.

longtest

A tool for testing the HTML render of the package description (part of `zest.releaser`).

mkrelease

A tool we use for releasing a new version (part of `jarn.mkrelease`).

5.2 Conventions

5.2.1 Line length

All Python code in this package should be PEP8 valid. However, we don't enforce the 80-char line length rule. It is encouraged to have your code formatted in 80-char lines, but somewhere it's just more readable to break this rule for a few characters. Long and descriptive test method names are a good example of this.

Note: Configuring your editor to display a line at 80th column helps a lot here and saves time.

Note: The line length rules also applies to non-python source files, such as documentation .rst files.

5.2.2 About imports

1. Don't use * to import *everything* from a module.
2. Don't use commas to import multiple stuff on a single line.
3. Don't use relative paths.

```
from collective.table.local import add_row
from collective.table.local import delete_rows
from collective.table.local import update_cell
```

instead of

```
from collective.table.local import *
from collective.table.local import add_row, delete_rows
from .local import update_cell
```

5.2.3 Sort imports

As another imports stylistic guide: Imports of code from other modules should always be alphabetically sorted with no empty lines between imports. The only exception to this rule is to keep one empty line between a group of `from x import y` and a group of `import y` imports.

```
from collective.table.tests.base import TableIntegrationTestCase
from plone.app.testing import login

import os
```

instead of

```
import os

from plone.app.testing import login
from collective.table.tests.base import TableIntegrationTestCase
```

5.2.4 Commit checklist

Before every commit you should:

- Run *Unit tests*.
- Run *Syntax validation*.
- Add an entry to *Changelog* (if applicable).
- Add/modify *Sphinx documentation* (if applicable).

5.2.5 Unit tests

Un-tested code is broken code.

For every feature you add to the codebase you must also add tests for it. Also write a test for every bug you fix to ensure it doesn't crop up again in the future.

You run tests like this:

```
$ bin/test
```

5.2.6 Syntax validation

All Python source code should be *PEP-8* valid and checked for syntax errors. Tool for checking this is *vvv*.

To validate your source code, run the following two commands:

```
$ bin/vvv src/spinnerchief
```

Note: It pays off to invest a little time to make your editor run *pep8* and *pyflakes* on a file every time you save that file. Saves lots of time in the long run.

5.2.7 Changelog

Feature-level changes to code are tracked inside `docs/HISTORY.txt`. Examples:

- added feature X
- removed Y
- fixed bug Z

Add an entry every time you add/remove a feature, fix a bug, etc.

5.2.8 Sphinx documentation

Un-documented code is broken code.

For every feature you add to the codebase you should also add documentation for it to `docs/`.

After adding/modifying documentation, re-build *Sphinx* and check how it is displayed:

```
$ bin/sphinxbuilder
$ open docs/html/index.html
```

Documentation is automatically generated from these source files every time you push your code to GitHub. The post-commit hook is handled by ReadTheDocs and the results are visible at http://readthedocs.org/docs/spinnerchief.

5.3 Releasing a new version

Releasing a new version of *spinnerchief* involves the following steps:

1. Create a git tag for the release.
2. Push the git tag upstream to GitHub.
3. Generate a distribution file for the package.
4. Upload the generated package to Python Package Index (PyPI).

5.3.1 Checklist

Before every release make sure that:

1. You have documented your changes in the `HISTORY.rst` file.
2. You have modified the version identifier in the `version.txt` to reflect the new release.
3. You have confirmed that the package description (generated from `README.rst` and others) renders correctly by running `bin/longtest`.
4. You have committed all changes to the git repository and pushed them upstream.
5. You have the working directory checked out at the revision you wish to release.

5.3.2 Actions

For help with releasing we use `jarn.mkreleaser`. It's listed as a dependency in `setup.py` and should already be installed in your local bin:

```
$ bin/mkrelease -d pypi -pq ./
```

Note: In order to push packages to PyPI you need to have the appropriate access rights to the package on PyPI and you need to configure your PyPI credentials in the `~/.pypirc` file, e.g.:

```
[distutils]
index-servers =
  pypi

[pypi]
username = fred
password = secret
```

5.3.3 Example

In the following example we are releasing version 0.1 of *spinnerchief*. The package has been prepared so that `version.txt` contains the version `0.1`, this change has been committed to git and all changes have been pushed upstream to GitHub:

```
# Check that package description is rendered correctly
$ bin/longtest

# Make a release and upload it to PyPI
$ bin/mkrelease -d pypi -pq ./
Releasing spinnerchief 0.1
Tagging spinnerchief 0.1
To git@github.com:niteoweb/spinnerchief.git
* [new tag]         0.1 -> 0.1
running egg_info
running sdist
warning: sdist: standard file not found: should have one of README, README.txt
running register
Server response (200): OK
running upload
warning: sdist: standard file not found: should have one of README, README.txt
Server response (200): OK
done
```

Note: Please ignore the sdist warning about README file above. PyPI does not depend on it and it's just a bug in setupools (reported and waiting to be fixed).

CHAPTER 6

Credits

- Development by NiteoWeb Ltd.
- Code and documentation snippets inspired by HexagonIT Oy.
- Similar package used as point-of-reference is thebestspinner by Peter Flood.

CHAPTER 7

Changelog

7.1 0.1 (2012-09-18)

- SpinnerChief class.
- Tests and documentation. [Matej Cotman]

CHAPTER 8

License (3-clause BSD)

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of NiteoWeb Ltd. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

CHAPTER 9

Indices and tables

- *genindex*
- *modindex*
- *search*

S