sphinxcontrib.spelling Documentation

Release 1.4

Doug Hellmann

January 26, 2014

`sphinxcontrib.spelling` is a spelling checker for Sphinx. It uses PyEnchant to produce a report showing misspelled words.

CHAPTER 1

Features

1. Supports multiple source languages using the standard enchant dictionaries.
2. Supports project-specific dictionaries for localized jargon and other terminology that may not appear in the global dictionaries.
3. Suggests alternatives to words not found in the dictionary, when possible.

CHAPTER 2

Details

2.1 Installation

2.1.1 Installing sphinxcontrib.spelling

1. Follow the instructions on the PyEnchant site to install **enchant** and then **PyEnchant**.
2. Install the extension with pip: `pip install sphinxcontrib-spelling`

2.1.2 Configuration

1. Add `'sphinxcontrib.spelling'` to the `extensions` list in `conf.py`.

```
extensions = [ 'sphinxcontrib.spelling' ]
```

2.2 Configuration Options

These options can be set in `conf.py` along with the other Sphinx configuration settings.

2.2.1 Input Options

`spelling_lang='en_US'` String specifying the language, as understood by PyEnchant and enchant. Defaults to `en_US` for US English.

`spelling_word_list_filename='spelling_wordlist.txt'` String specifying a file containing a list of words known to be spelled correctly but that do not appear in the language dictionary selected by `spelling_lang`. The file should contain one word per line. Refer to the PyEnchant tutorial for details.

2.2.2 Output Options

`spelling_show_suggestions=False` Boolean controlling whether suggestions for misspelled words are printed. Defaults to False.

2.2.3 Word Filters

Enable or disable the built-in filters to control which words are returned by the tokenizer to be checked.

spelling_ignore_pypi_package_names=False Boolean controlling whether words that look like package names from PyPI are treated as spelled properly. When `True`, the current list of package names is downloaded at the start of the build and used to extend the list of known words in the dictionary. Defaults to `False`.

spelling_ignore_wiki_words=True Boolean controlling whether words that follow the CamelCase conventions used for page names in wikis should be treated as spelled properly. Defaults to `True`.

spelling_ignore_acronyms=True Boolean controlling treatment of words that appear in all capital letters, or all capital letters followed by a lower case `s`. When `True`, acronyms are assumed to be spelled properly. Defaults to `True`.

spelling_ignore_python_builtins=True Boolean controlling whether names built in to Python should be treated as spelled properly. Defaults to `True`.

spelling_ignore_importable_modules=True Boolean controlling whether words that are names of modules found on `sys.path` are treated as spelled properly. Defaults to `True`.

spelling_filters=[] List of filter classes to be added to the tokenizer that produces words to be checked. The classes should be derived from `enchant.tokenize.Filter`. Refer to the PyEnchant tutorial for examples.

2.2.4 Private Dictionaries

There are two ways to provide a list of known good words. The `spelling_word_list_filename` option (described above) specifies the name of a plain text file containing one word per line. All of the words in the file are assumed to be spelled correctly and may appear in any part of the document being processed.

The `spelling` directive can be used to create a list of words known to be spelled correctly within a single file. For example, if a document refers to a person or project by name, the name can be added to the list of known words for just that document.

```
.. spelling::

   Docutils
   Goodger
```

2.2.5 Custom Word Filters

The PyEnchant tokenizer supports a "filtering" API for processing words from the input. Filters can alter the stream of words by adding, replacing, or dropping values.

New filters should be derived from `enchant.tokenize.Filter` and implement either the `_split()` method (to add or replace words) or `_skip()` (to treat words as being spelled correctly). For example, this `AcronymFilter` skips words that are all uppercase letters or all uppercase with a trailing lowercase "s".

```
class AcronymFilter(Filter):
    """If a word looks like an acronym (all upper case letters),
    ignore it.
    """

    def _skip(self, word):
        return (word == word.upper() # all caps
                or
                # pluralized acronym ("URLs")
```

```
                (word[-1].lower() == 's'
                 and
                 word[:-1] == word[:-1].upper()
                 )
                )
```

To be used in a document, the custom filter needs to be installed somewhere that Sphinx can import it while processing the input files. The Sphinx project's `conf.py` then needs two changes.

1. Import the filter class.
2. Add the filter class to the `spelling_filters` configuration variable.

```
from mymodule import MyFilter

spelling_filters = [MyFilter]
```

See also:

- Creating a Spelling Checker for reStructuredText Documents
- PyEnchant tutorial

2.3 Running

To process a document with the spell checker, use `sphinx-build` and specify `spelling` as the builder name using the `-b` option. The output includes the headings from the document and any misspelled words. If suggestions are enabled, they are shown on the same line as the misspelling. A log of all the words not found in the dictionary is saved to the file `spelling/output.txt` under the build directory.

```
$ make spelling
sphinx-build -b spelling -d build/doctrees   source build/spelling
Running Sphinx v1.0.7
Initializing Spelling Checker
loading pickled environment... done
building [spelling]: all documents
updating environment: 1 added, 2 changed, 0 removed
reading sources... [ 33%] history
reading sources... [ 66%] index
reading sources... [100%] run

looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [ 20%] developers
writing output... [ 40%] history
writing output... [ 60%] index
writing output... [ 80%] install
(line  28) PyEnchant ["Penchant"]
writing output... [100%] run

Spelling checker messages written to ./docs/build/spelling/output.txt
build finished with problems.
make: *** [spelling] Error 1
```

2.4 Developers

If you would like to contribute to sphinxcontrib.spelling directly, these instructions should help you get started. Patches, bug reports, and feature requests are all welcome through the BitBucket site. Contributions in the form of patches or pull requests are easier to integrate and will receive priority attention.

2.4.1 Building Documentation

The documentation for sphinxcontrib.spelling is written in reStructuredText and converted to HTML using Sphinx. The build itself is driven by make. You will need the following packages in order to build the docs:

- Sphinx
- docutils
- sphinxcontrib.spelling

Once all of the tools are installed into a virtualenv using pip, run `make html` to generate the HTML version of the documentation.

2.5 Release History

1.4

- Fixed detection of builtins under PyPy, contributed by Hong Minhee (https://bitbucket.org/dahlia).

1.3

- Handle text nodes without parents. (#19)
- Include the input document name in the console output.
- Use the Sphinx wrapper for registering a directive.

1.2

- Add the document name to the messages showing the contents of a local dictionary created by the `spelling` directive.
- Add title nodes to the list of node types checked for spelling. Resolves issue #17.
- Add test/test_wordlist.txt to the manifest so it is included in the source distribution and the tests will pass. Resolves issue #17.
- Documentation patch from Hank Gay.

1.1.1

- Fix initialization so the per-document filters work even if no `spelling` directive is used.

1.1

- Add an option treat the names of packages on PyPI as spelled properly.
- Add an option to treat CamelCase names as spelled properly.
- Add an option to treat acronyms as spelled properly.
- Add an option to treat Python built-ins as spelled properly.
- Add an option to treat names that can be found as modules as spelled properly.

- Add an option to let the user provide a list of other filter classes for the tokenizer.
- Add `spelling` directive for passing local configuration settings to the spelling checker. This version allows setting a list of words known to be spelled correctly.

1.0

- Re-implement using just a Builder, without a separate visitor class.
- Show the file and line number of any words not appearing in the dictionary, instead of the section title.
- Log the file, line, and unknown words as the documents are processed.

0.2

- Warn but otherwise ignore unknown node types.

0.1

- First public release.