Python GTK Spellcheck Documentation

Release 4.0

Maximilian Köhl & Carlos Jenkins

June 27, 2016

Contents

A simple but quite powerful spellchecking library written in pure Python for Gtk based on Enchant. It supports PyGObject as well as PyGtk for Python 2 and 3 with automatic switching and binding detection. For automatic translation of the user interface it can use Gedit's translation files.

CHAPTER 1

Features

- localized names of the available languages
- supports word, line and multiple line ignore regular expressions
- supports ignore custom tags on GtkTextBuffer
- enable and disable of spellchecking with preferences memory
- supports hotswap of GtkTextBuffers
- PyGObject and PyGtk compatible with automatic detection
- Python 2 and 3 supportas Enchant, support for Hunspell (LibreOffice) and Aspell (GNU) dictionaries
- extract dictionaries out of LibreOffice extension files
- legacy API for Python GtkSpell

CHAPTER 2

API Reference

class gtkspellcheck.spellcheck.**SpellChecker**(*view*, *language='en'*, *prefix='gtkspellchecker'*, *collapse=True*, *params={}*)

Main spellchecking class, everything important happens here.

Parameters

- **view** – GtkTextView the SpellChecker should be attached to.
- **language** – The language which should be used for spellchecking. Use a combination of two letter lower-case ISO 639 language code with a two letter upper-case ISO 3166 country code, for example en_US or de_DE.
- **prefix** – A prefix for some internal GtkTextMarks.
- **collapse** – Enclose suggestions in its own menu.
- **params** – Dictionary with Enchant broker parameters that should be set e.g. *enchant.myspell.dictionary.path*.

languages

A list of supported languages.

exists(*language*)

Checks if a language exists.

Parameters **language** – language to check

add_to_dictionary(*word*)

Adds a word to user's dictionary.

Parameters **word** – The word to add.

append_filter(*regex*, *filter_type*)

Append a new filter to the filter list. Filters are useful to ignore some misspelled words based on regular expressions.

Parameters

- **regex** – The regex used for filtering.
- **filter_type** – The type of the filter.

Filter Types:

SpellChecker.FILTER_WORD: The regex must match the whole word you want to filter. The word separation is done by Pango's word separation algorithm so, for example, urls won't work here because they are split in many words.

SpellChecker.FILTER_LINE: If the expression you want to match is a single line expression use this type. It should not be an open end expression because then the rest of the line with the text you want to filter will become correct.

SpellChecker.FILTER_TEXT: Use this if you want to filter multiline expressions. The regex will be compiled with the *re.MULTILINE* flag. Same with open end expressions apply here.

append_ignore_tag(*tag*)
Appends a tag to the list of ignored tags. A string will be automatic resolved into a tag object.

Parameters **tag** – Tag object or tag name.

buffer_initialize()
Initialize the GtkTextBuffer associated with the GtkTextView. If you have associated a new GtkTextBuffer with the GtkTextView call this method.

check_range(*start*, *end*, *force_all=False*)
Checks a specified range between two GtkTextIters.

Parameters

- **start** – Start iter - checking starts here.
- **end** – End iter - checking ends here.

disable()
Disable spellchecking.

enable()
Enable spellchecking.

enabled
Enable or disable spellchecking.

ignore_all(*word*)
Ignores a word for the current session.

Parameters **word** – The word to ignore.

language
The language used for spellchecking.

recheck()
Rechecks the spelling of the whole text.

remove_filter(*regex*, *filter_type*)
Remove a filter from the filter list.

Parameters

- **regex** – The regex which used for filtering.
- **filter_type** – The type of the filter.

remove_ignore_tag(*tag*)
Removes a tag from the list of ignored tags. A string will be automatic resolved into a tag object.

Parameters **tag** – Tag object or tag name.

class gtkspellcheck.spellcheck.**NoDictionariesFound**
There aren't any dictionaries installed on the current system so spellchecking could not work in any way.

class gtkspellcheck.spellcheck.**NoGtkBindingFound**
Could not find any loaded Gtk binding.

pylocales.**code_to_name**(*code*, *separator='_'*)

Get the human readable and translated name of a language based on it's code.

Parameters

- **code** – the code of the language (e.g. de_DE, en_US)
- **target** – separator used to separate language from country

Return type human readable and translated language name

gtkspellcheck.oxt_extract.**extract**(*filename*, *target*, *override=False*)

Extract Hunspell dictionaries out of LibreOffice `.oxt` extensions.

Parameters

- **filename** – path to the `.oxt` extension
- **target** – path to extract Hunspell dictionaries to
- **override** – override existing files in the target directory

Return type list of the extracted dictionaries

This function extracts the Hunspell dictionaries (`.dic` and `.aff` files) from the given `.oxt` extension found to `target`.

Extensions could be found at:

http://extensions.services.openoffice.org/dictionary

gtkspellcheck.oxt_extract.**batch_extract**(*oxt_path*, *extract_path*, *override=False*, *move_path=None*)

Uncompress, read and install LibreOffice `.oxt` dictionaries extensions.

Parameters

- **oxt_path** – path to a directory containing the `.oxt` extensions
- **extract_path** – path to extract Hunspell dictionaries files to
- **override** – override already existing files
- **move_path** – optional path to move the `.oxt` files after processing

Return type generator over all extensions, yielding result, extension name, error, extracted dictionaries and translated error message - result would be `BATCH_SUCCESS` for success, `BATCH_ERROR` if some error happened or `BATCH_WARNING` which contain some warning messages instead of errors

This function extracts the Hunspell dictionaries (`.dic` and `.aff` files) from all the `.oxt` extensions found on `oxt_path` directory to the `extract_path` directory.

Extensions could be found at:

http://extensions.services.openoffice.org/dictionary

In detail, this functions does the following:

1. find all the `.oxt` extension files within `oxt_path`
2. open (unzip) each extension
3. find the dictionary definition file within (*dictionaries.xcu*)
4. parse the dictionary definition file and locate the dictionaries files
5. uncompress those files to `extract_path`

By default file overriding is disabled, set `override` parameter to True if you want to enable it. As additional option, each processed extension can be moved to `move_path`.

Example:

```
for result, name, error, dictionaries, message in oxt_extract.batch_extract(...):
    if result == oxt_extract.BATCH_SUCCESS:
        print('successfully extracted extension "{}"'.format(name))
    elif result == oxt_extract.BATCH_ERROR:
        print('could not extract extension "{}"'.format(name))
        print(message)
        print('error {}'.format(error))
    elif result == oxt_extract.BATCH_WARNING:
        print('warning during processing extension "{}"'.format(name))
        print(message)
        print(error)
```

class gtkspellcheck.oxt_extract.**BadXml**
The XML dictionary registry is not valid XML.

class gtkspellcheck.oxt_extract.**BadExtensionFile**
The extension has a wrong file format, should be a ZIP file.

class gtkspellcheck.oxt_extract.**ExtractPathIsNoDirectory**
The given *extract_path* is no directory.

CHAPTER 3

Development

Development happens at GitHub.

```
git clone git://github.com/koehlma/pygtkspellcheck.git
```

Download last sources in a ZIP or Tarball file.

CHAPTER 4

Website

Checkout the official project website for additional information.

CHAPTER 5

Examples

CHAPTER 6

License

PyGtkSpellcheck is released under GPLv3 or at your opinion any later version.

A

B

C

D

E

I

L

N

R

S