klocmod

Nov 13, 2019

Contents:

- genindex

CHAPTER 1

Description and reference

Screw you, gettext! I don't wanna bother of compiling strings into binary files!

This module provides a very simple, suboptimal way for localizing your scripts, bots or applications. The advantage is its simplicity: to supply some sets of different string literals for different languages, you just need a simple JSON, YAML or INI file (or even a dict) fed to the library. After that, the only thing you should take care of is to get an instance of the dictionary for a specific language and extract messages from it by key values.

All you mostly want is the *LocalizationsContainer* class. In particular, its static method *LocalizationsContainer.from_file()* that reads a localization file and returns an instance of the factory. The factory is supposed to produce instances of the *LanguageDictionary* class. Most likely, you will encounter instances of its subclass – the *SpecificLanguageDictionary* class (the base class is only used as a fallback that returns passed key values back).

1.1 Installation

```
# basic installation
pip install klocmod
# or with YAML files support enabled
pip install klocmod[YAML]
```

1.1.1 Examples of localization files

1.2 JSON (language first)

```
{
  "en": {
    "yes": "yes",
    "no": "no"
  },
```

(continues on next page)

(continued from previous page)

```
  "ru-RU": {
    "yes": "",
    "no": ""
  }
}
```

1.3 JSON (phrase first)

```
{
  "yes": {
    "en": "yes",
    "ru-RU": ""
  },
  "no": {
    "en": "no",
    "ru-RU": ""
  }
}
```

1.4 INI

```
[DEFAULT]
yes = yes
no = no

[ru-RU]
yes =
no =
```

1.5 YAML

Requires an extra dependency: *PyYAML*.

```
# language first
en:
  yes: yes
  no: no
ru-RU:
  yes:
  no:
---
# phrase first
yes:
  en: yes
  ru-RU:
no:
  en: no
  ru-RU:
```

1.5.1 Code example

```
from klocmod import LocalizationsContainer

localizations = LocalizationsContainer.from_file("localization.json")
ru = localizations.get_lang("ru")
# or
en = localizations.get_lang()     # get default language
# then
print(ru['yes'])     # output:
# alternative ways to get a specific phrase:
localizations.get_phrase("ru-RU", "no")
localizations['ru-RU']['no']
```

class klocmod.**LanguageDictionary**(*name: str*)

The base class for dict-like objects containing phrases for a particular language. Usually used as a fallback since just returns keys back (printing a warning into the log, of course).

You shouldn't instantiate objects of this class on your own. Use the *LocalizationsContainer* class instead.

The class supports equality testing based on the *name* property. Note that if the other object is not an instance of the *LanguageDictionary* class, a `TypeError` exception will be thrown.

name

The name of the locale ('ru-RU', 'en', etc.)

class klocmod.**SpecificLanguageDictionary**(*name: str, primary_dict: Dict[str, str], spare_dict: klocmod.LanguageDictionary*)

A concrete implementation of *LanguageDictionary* that consists of two dicts: one of them is considered as primary one and the other as spare. When you're trying to get some localized phrase by a key, the primary dict is used for searching first. If there is no such a key there, the search continues in the spare dict. Finally, if there is no such a phrase, the key itself is returned by the base class.

Using instances of this class it's possible to make chains of language dictionaries. For example, you can create the following chain of searching: `fr-CA -> fr -> en (default language) -> fallback`

In fact, this approach is used for localization files parsed by the `LocalizationContainer` class.

All missing phrases will be present in the log.

Parameters

- **name** – a language tag
- **primary_dict** – an actual dict of localized strings
- **spare_dict** – an instance of *LanguageDictionary* that is used as a fallback

class klocmod.**LocalizationsContainer**(*dct: Dict[str, Dict[str, str]], default_lang: str = 'en'*)

A factory of *LanguageDictionary* instances. Call the `LocalizationContainer.from_file()` static method to get the instance of the container. Then you can use the `LocalizationContainer.get_lang()` method to create instances of specific languages (*LanguageDictionary*).

Parameters

- **dct** – see examples in the module description
- **default_lang** – what language key will be used as a fallback

classmethod from_file(*path: Union[pathlib.PurePath, str], default_lang: str = 'en'*) → klocmod.LocalizationsContainer

A factory method that reads a given file and returns an instance of the *LocalizationsContainer* class.

Currently supported formats are **JSON**, **INI** and **YAML** (if *PyYAML* is installed).

Parameters

- **path** – a path to the localization file
- **default_lang** – what language key will be used as a fallback

Returns an instance of *LocalizationsContainer*

get_lang(*lang_tag: str = None*) → klocmod.LanguageDictionary

The same as *get_language()* but takes either hyphen-separated and underscore-separated language tags as a one entry. Returns the default language if the desired language doesn't present in the localization container (or if `lang_tag` is *None*).

Parameters **lang_tag** – language tags such as "en-US", "en_AU" or just "en"

Returns an instance of *LanguageDictionary*

get_language(*language: str*, *country: str = None*) → klocmod.LanguageDictionary

Returns a set of phrases for some certain language.

Parameters

- **language** – 2 character code of the language
- **country** – 2 character code of the country (may be omitted)

Returns an instance of *LanguageDictionary*

get_phrase(*lang_tag: str*, *key: str*) → str

A shortcut for *get_lang()* that let you retrieve only a one phrase at once, not a whole dictionary.

Parameters

- **lang_tag** – "en-US", "en_AU" or just "en"
- **key** – a key identified the phrase in the dictionary

Returns the locale-specific phrase

exception klocmod.**InvalidLocalizationFileError**(*message: str, file_path: Union[pathlib.PurePath, str], nested_exc: Exception = None*)

An exception that is thrown when any error occurred while parsing some localization file.

Parameters

- **message** – a human-readable message describing the error
- **file_path** – a path to the localization file
- **nested_exc** – another exception that caused this one, if present

file_path

A full path to the localization file caused the error.

filename

The name of the localization file caused the error.

nested_exception

Another exception that caused this one.

Python Module Index

k

Index