

linguify manual



Abstract

linguify is a package for loading strings for different languages easily.

Version: 0.4.1

Authors: [jomaway](#) + community contributions

License: MIT

Contents

Usage	2
Basic Example	2
Information for package authors.	3
Fluent support	3
Contributing	4
Reference	5
Linguify reference	5

This manual shows a short example for the usage of the `linguify` package inside your document. If you want to **include linguify into your package** make sure to read the section for package authors .

Usage

Basic Example

Load language data file: → See database section for content of `lang.toml`

```
#set-database(toml("lang.toml"))
```

Example input:

```
#set text(lang: "LANG")
#smallcaps(linguify("abstract"))
=== #linguify("title")
```

Test: `#linguify("test")`

Lang	Output
en	ABSTRACT A simple linguify example Test: testing
de	ZUSAMMENFASSUNG Ein einfaches Linguify Beispiel Test: testen
es	RESUMEN Un ejemplo sencillo de linguify Test: testing <i>Info: The key «test» is missing in the «es» language section, but as we specified a default-lang in the <code>conf</code> it will display the entry inside the specified language section, which is «en» in our case. To disable this behavior delete the <code>default-lang</code> entry from the <code>lang.toml</code>.</i>
CZ	ABSTRACT A simple linguify example Test: testing <i>Info: As the lang data does not contain a section for “cz” this entire output will fallback to the default-lang. To disable this behavior delete the <code>default-lang</code> entry from the <code>lang.toml</code>.</i>

Database

The content of the `lang.toml` file, used in the example above looks like this.

```
[conf]
default-lang = "en"

[lang.en]
title = "A simple linguify example"
abstract = "Abstract"
test = "testing"

[lang.de]
title = "Ein einfaches Linguify Beispiel"
abstract = "Zusammenfassung"
test = "testen"

[lang.es]
title = "Un ejemplo sencillo de linguify"
abstract = "Resumen"

[lang.fr]
title = "Un exemple simple de linguify"
abstract = "résumé"
```

Information for package authors.

As the database is stored in a `typst` state, it can be overwritten. This leads to the following problem. If you use *linguify* inside your package and use the `set_database()` function it will probably work like you expect. But if a user imports your package and uses *linguify* for their own document as well, he will overwrite the your database by using `set_database`. Therefore it is recommend to use the `from` argument in the `linguify` function to specify your database directly.

Example:

```
// Load data
#let lang_data = toml("lang.toml")

// Usage
#linguify("key", from: lang_data)
```

This makes sure the end user still can use the global database provided by *linguify* with `set_database()` and calling.

→ Have a look at the [gentle-clues](#) package for a real live example.

Fluent support

Thanks to [sjfhsjfh](#) we have fluent support.

Fluent is “a localization system for natural-sounding translations.” ([Project Fluent](#))

Heres a simple example of how to use the `linguify` package to load translations from fluent files, which are kept in `L10n` directory and named with the language code, e.g. `en.ftl` and `zh.ftl`.

```
// my-document.typ
#import "@preview/linguify:0.4.0": *
// Define the languages you have files for.
#let languages = ("en", "zh")
```

Folder structure

```
my-project
├── L10n
│   └── en.ftl
```

```
#set_database(eval(load_ftl_data("./L10n",
languages)))
```

```
// Use linguify like described above.
= #linguify("title")
```

```
#set text(lang: "zh")
= #linguify("title")
```

```
// Args are supported as well.
#linguify("hello", lang: "en", args:
("name": "Alice & Bob",))
```

You have to maintain the language list used in database initialization since Typst currently does not list files in a directory. Of course, you can use an external file to store the language list and load it in the script if it is necessary.

Store config inside a `lang.toml` file. Load config inside your document.

```
[conf]
default-lang = "en"
data-type = "ftl"

[ftl]
languages = ["en", "de"]
path = "./L10n"
```

```
[ftl.args]
name = "Lore"
```

```
[lang]
```

```
|   └─ zh.ftl
|
└─ my-document.typ
```

Example for `en.ftl`

```
title = A linguify example - with Fluent
abstract = Abstract
hello = Hello, {$name}!
```

```
#let data = toml("lang.toml")

#for lang in data.ftl.languages {
  let lang_section = read(data.ftl.path + "/" + lang
+ ".ftl")
  data.lang.insert(lang, lang_section)
}
```

```
#set_database(data)
#linguify("hello")
```

→ prints `Hello, Lore!`

Contributing

If you would like to integrate a new i18n solution into *linguify*, you can set the `conf.data_type` described in the database section . And then add implementation in the `get-text` function for your data type.

Reference

Linguify reference

set-database

Set the default linguify database

The data must contain at least a lang section like described at [database](#).

Parameters

```
set-database(data: dictionary) -> content (state-update)
```

data dictionary

the database which will be set to [database](#)

reset-database

Clear current database

Parameters

```
reset-database() -> content (state-update)
```

get_text

Get a value from a L10n data dictionary.

Parameters

```
get_text(  
  src: dictionary ,  
  key: string ,  
  lang: string ,  
  mode: string ,  
  args  
) -> The value for the key in the dictionary. If the key does not exist, `none` is returned.
```

src dictionary

The dictionary to get the value from.

key string

The key to get the value for.

lang string

The language to get the value for.

mode `string`

The data structure of src

Default: `"dict"`

linguify

fetch a string in the required language. provides context for `_linguify` function which implements the logic part.

Parameters

```
linguify(  
    key: string ,  
    from: dictionary ,  
    lang: string ,  
    default: any ,  
    args  
) -> content
```

key `string`

The key at which to retrieve the item.

from `dictionary`

database to fetch the item from. If auto linguify's global database will used.

Default: `auto`

lang `string`

the language to look for, if auto use `context text.lang` (default)

Default: `auto`

default `any`

A default value to return if the key is not part of the database.

Default: `auto`

database

None or dictionary of the following structure:

- `conf`

- `data_type` (string): The type of data structure used for the database. If not specified, it defaults to `dict` structure.
- `default-lang` (string): The default language to use as a fallback if the key in the preferred language is not found.
- ...
- `lang`
 - `en` : The English language section.
 - ...