Package 'stranslate'

July 23, 2025

Type Package

Title Simple Translation Between Different Languages

Version 0.1.3

Maintainer Sigbert Klinke <sigbert@hu-berlin.de>

Description Message translation is often managed with 'po' files and the 'gettext' programme, but sometimes another solution is needed. In contrast to 'po' files, a more flexible approach is used as in the Fluent <https://projectfluent.org/> project with R Markdown snippets. The key-value approach allows easier handling of the translated messages.

URL https://github.com/sigbertklinke/stranslate (development version)

VignetteBuilder knitr

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

Suggests rmarkdown

Imports crayon, knitr, stringr

NeedsCompilation no

Author Sigbert Klinke [aut, cre]

Repository CRAN

Date/Publication 2024-01-28 10:10:02 UTC

Contents

`allFunctions` *allFunctions*

Description

Returns the names of all functions currently available.

Usage

```
allFunctions()
```

Value

character vector of functions names

References

https://stackoverflow.com/questions/4267744/is-there-a-way-to-get-a-vector-with-the-name-of-all-functions-that-one-could-use

Examples

```
allFunctions()
```

`findText` *findText*

Description

Extract all texts, `getMsg`, or `setMsg` calls in the R files given. Note that the `'` and `"` are part of the string, thus `'ROUND'` has a length of 7!

Usage

```
findText(files, pattern = "\\w+{2,}")
```

Arguments

`files`	character: names of R files to analyse
`pattern`	character: pattern to match a string (default: `"\\w+{2,}"`)

Value

a data frame with the columns

- `file` with file name
- `line` the line number
- `type` if a string constant (`STR_CONST`), a `getMsg` (`GETMSG`), or a `setMsg` (`SETMSG`) was found
- `text` the text of the string constant or the key used in `getMsg` or `setMsg`

Examples

```
findText(system.file("messages", "messages.R", package="stranslate"))
```

`getMsg` *getMsg*

Description

Returns a message. The first parameter must be the key to the message. For details read the vignette `vignette("stranslate")`.

Usage

```
getMsg(
  ...,
  .domain = getOption("stranslate.domain"),
  .lang = getOption("stranslate.lang")
)
```

Arguments

`...`	parameter(s) given to the function
`.domain`	character: domain name (default: `getOption("stranslate.domain")`)
`.lang`	character: language to use (default: `getOption("stranslate.lang")`)

Value

the (translated) message

Examples

```
# without a parameter
getMsg("DOMAIN_UNIQUE", .domain="stranslate", .lang="en")
getMsg('DOMAIN_UNIQUE', .domain="stranslate", .lang="de")
getMsg(DOMAIN_UNIQUE, .domain="stranslate")
# with a parameter
getMsg(LANGUAGE="english", .domain="stranslate", .lang="en")
getMsg(LANGUAGE="deutsch", .domain="stranslate", .lang="de")
# which system language is used?
getMsg(LANGUAGE=Sys.getenv("LANG"), .domain="stranslate")
```

isName *isName*

Description

Checks if a `txt` consists of valid name(s):

- A name must start with a letter and can be a combination of letters, digits, period(.) and underscore(_).
- Reserved words cannot be used as a name (TRUE, FALSE, NULL, if...)

Usage

```
isName(txt)
```

Arguments

`txt`	character: name(s) to check

Value

a logical vector

Examples

```
isName("?plot")
isName(".default")
```

language *language*

Description

Returns which loaded language will be used for finding a message depending on `.domain`.

Usage

```
language(
  lang = getOption("stranslate.lang"),
  available.languages = NULL,
  .domain = NULL
)
```

Arguments

lang	character: a language code, e.g. from ISO-639
available.languages	character: names of languages (default: NULL). If NULL then list of loaded languages is used.
.domain	character: domain names (default: NULL)

Value

character of languages use to find a message

Examples

```
print(options("stranslate.lang"))   # current default language
language("de_AT", c("de", "de_IT")) # request austrian german
#
loadMsg(system.file("messages", "messages.txt", package = "stranslate"))
language("de_AT")                   # request austrian german
language("tlh")                     # request klingon, not available -> "en"
```

---

listMsg *listMsg*

Description

Lists all keys in all languages in a specific domain or all domains (default).

Usage

```
listMsg(.domain = NULL)
```

Arguments

.domain	character: domains(s) to list (default: NULL)

Value

nothing

Examples

```
listMsg("stranslate")
listMsg()
# load some more text
loadMsg(system.file("messages/messages.txt", package="stranslate"), .overwrite=TRUE) # avoid warning
listMsg()
```

loadMsg *loadMsg*

Description

Loads translated messages from one (or more) file into package environment. For details read the vignette `vignette("stranslate")`.

Usage

```
loadMsg(
  files,
  .domain = getOption("stranslate.domain"),
  .lang = "en",
  .silent = TRUE,
  .overwrite = FALSE
)
```

Arguments

`files`	character: names of text file(s) with translated messages
`.domain`	character: domain namesd (default: `getOption("stranslate.domain")`)
`.lang`	character: default language (default: 'en')
`.silent`	logical: should outputs be displayed during the loading process and then `listMsg()` be called?
`.overwrite`	logical: should keys be overwritten (default: `FALSE`)

Value

invisibly the current message environment

Examples

```
# note "messages/messages.txt" contains only english and german ;)
loadMsg(system.file("messages/messages.txt", package="stranslate"), .overwrite=TRUE) # avoid warning
# english
getMsg(ROUND=0, .lang="en")
getMsg(ROUND=1, .lang="en")
getMsg(ROUND=2, .lang="en")
# english
getMsg(ROUND=0, .lang="de")
getMsg(ROUND=1, .lang="de")
getMsg(ROUND=2, .lang="de")
# default language or if not available then english
getMsg(ROUND=0)
getMsg(ROUND=1)
getMsg(ROUND=2)
```

setLang *setLang*

Description

Sets the language and the domain.

Usage

```
setLang(.lang = "en", .domain = getOption("stranslate.domain"))
```

Arguments

.lang	character: language (default: "en")
.domain	character: domain (default: getOption("stranslate.domain"))

Value

invisibly the language set

Examples

```
setLang('tlh')  # use klingon now ;)
```

setMsg *setMsg*

Description

Sets for a key the default message and other optional messages. The first argument specifies the key and the default message. Further named arguments give the message if the key is the same as the name,

Usage

```
setMsg(..., .silent = TRUE, .overwrite = FALSE)
```

Arguments

...	first argument is a key and the default message, further named arguments give optional messages
.silent	logical: should the key shown during the process
.overwrite	logical: should keys be overwritten (default: FALSE)

Value

returns invisibly the key

Examples

```
setLang("de", .domain="round")
# If ROUND=0 then getMsg returns 'Runden Sie ihr Ergebnis auf eine ganze Zahl'
# If ROUND=1 then getMsg returns 'Runden Sie ihr Ergebnis auf eine Nachkommastelle'
# Otherwise getMsg retuns 'Runden Sie ihr Ergebnis auf `r ROUND` Nachkommastellen'
setMsg(ROUND='Runden Sie ihr Ergebnis auf `r ROUND` Nachkommastellen',
       '0'='Runden Sie ihr Ergebnis auf eine ganze Zahl',
       '1'='Runden Sie ihr Ergebnis auf eine Nachkommastelle')
getMsg(ROUND=0, .lang="de", .domain="round")
getMsg(ROUND=1, .lang="de", .domain="round")
getMsg(ROUND=2, .lang="de", .domain="round")
```

Index