Package 'shiny.i18n'

January 16, 2023

Title Shiny Applications Internationalization

Version 0.3.0

Description It provides easy internationalization of Shiny applications. It can be used as standalone translation package to translate reports, interactive visualizations or graphical elements as well.

Depends R (>= 3.3.0)

Imports yaml, jsonlite, methods, stringr, R6, glue, shiny, rstudioapi, utils

License MIT + file LICENSE

Encoding UTF-8

URL `https://appsilon.github.io/shiny.i18n/`, `https://github.com/Appsilon/shiny.i18n`

BugReports `https://github.com/Appsilon/shiny.i18n/issues`

RoxygenNote 7.2.1

Suggests covr, googleLanguageR, knitr, lintr, rcmdcheck, rmarkdown, spelling, testthat, withr, quarto

Language en-US

NeedsCompilation no

Author Jakub Nowicki [cre, aut], Dominik Krzemiński [aut], Krystian Igras [aut], Jakub Sobolewski [aut], Appsilon Sp. z o.o. [cph]

Maintainer Jakub Nowicki <`opensource+kuba@appsilon.com`>

Repository CRAN

Date/Publication 2023-01-16 10:40:02 UTC

R topics documented:

`create_translation_file` *Create translation file*

Description

Auxiliary shiny.i18n function that searches for all key expressions (e.g. surrounded by `i18n$t()` tag in the script).

Usage

```
create_translation_file(path, type = "json", handle = "i18n", output = NULL)
```

Arguments

`path`	character with path of the file that needs to be inspected for key translations
`type`	type of the example output file with translations, either "json" or "csv"
`handle`	name of `Translator` object within script from `path`
`output`	if NULL (default) the output will be saved with a default file name ("translation.json" for JSON and "translate_lang.csv" for CSV)

`init_i18n` *Creates new i18n Translator object*

Description

Creates new i18n Translator object

Usage

```
init_i18n(
  translation_csvs_path = NULL,
  translation_json_path = NULL,
  translation_csv_config = NULL,
  automatic = FALSE
)
```

Arguments

`translation_csvs_path`	character with path to folder containing csv translation files. See more in Details.
`translation_json_path`	character with path to JSON translation file. See more in Details.
`translation_csv_config`	character with path to configuration file for csv option.
`automatic`	logical flag, indicating if i18n should use an automatic translation API.

Value

i18n object

Examples

```
## Not run:
 i18n <- init_i18n(translation_csvs_path = "../csvdata/")
 i18n <- init_i18n(translation_json_path = "translations.json")

## End(Not run)
```

`translate_with_google_cloud`
This provides functions for automatic translations with online APIs Translate with Google cloud

---

Description

This is wrapper for `gl_translate` function from `googleLanguageR` package.

Usage

```
translate_with_google_cloud(txt_to_translate, target_lang)
```

Arguments

`txt_to_translate`	character with text to translate
`target_lang`	character with language code

`Translator` *Translator R6 Class*

Description

Translator R6 Class

Translator R6 Class

Details

This creates shinny.i18n Translator object used for translations. Now you can surround the pieces of the text you want to translate by one of the translate statements (ex.: `Translator$t("translate me")`). Find details in method descriptions below.

Methods

Public methods:

- `Translator$new()`
- `Translator$get_languages()`
- `Translator$get_translations()`
- `Translator$get_key_translation()`
- `Translator$get_translation_language()`
- `Translator$translate()`
- `Translator$t()`
- `Translator$set_translation_language()`
- `Translator$parse_date()`
- `Translator$parse_number()`
- `Translator$automatic_translate()`
- `Translator$at()`
- `Translator$use_js()`
- `Translator$clone()`

Method `new()`**:** Initialize the Translator with data

Usage:

```
Translator$new(
  translation_csvs_path = NULL,
  translation_json_path = NULL,
  translation_csv_config = NULL,
  separator_csv = ",",
  automatic = FALSE
)
```

Arguments:

`translation_csvs_path` character with path to folder containing csv translation files. Files must have "translation_" prefix, for example: `translation_<LANG-CODE>.csv`.

`translation_json_path` character with path to JSON translation file. See more in Details.
`translation_csv_config` character with path to configuration file for csv option.
`separator_csv` separator of CSV values (default ",")
`automatic` logical flag, indicating if i18n should use an automatic translation API.

Method `get_languages()`**:** Get all available languages

Usage:

```
Translator$get_languages()
```

Method `get_translations()`**:** Get whole translation matrix

Usage:

```
Translator$get_translations()
```

Method `get_key_translation()`**:** Get active key translation

Usage:

```
Translator$get_key_translation()
```

Method `get_translation_language()`**:** Get current target translation language

Usage:

```
Translator$get_translation_language()
```

Method `translate()`**:** Translates 'keyword' to language specified by 'set_translation_language'

Usage:

```
Translator$translate(keyword, session = shiny::getDefaultReactiveDomain())
```

Arguments:

`keyword` character or vector of characters with a word or expression to translate
`session` Shiny server session (default: current reactive domain)

Method `t()`**:** Wrapper to `translate` method.

Usage:

```
Translator$t(keyword, session = shiny::getDefaultReactiveDomain())
```

Arguments:

`keyword` character or vector of characters with a word or expression to translate
`session` Shiny server session (default: current reactive domain)

Method `set_translation_language()`**:** Specify language of translation. It must exist in 'languages' field.

Usage:

```
Translator$set_translation_language(transl_language)
```

Arguments:

`transl_language` character with a translation language code

Method `parse_date()`**:** Parse date to format described in 'cultural_date_format' field in config.

Usage:

`Translator$parse_date(date)`

Arguments:

`date` date object to format

Method `parse_number()`**:** Numbers parser. Not implemented yet.

Usage:

`Translator$parse_number(number)`

Arguments:

`number` numeric or character with number

Returns: character with number formatting

Method `automatic_translate()`**:** Translates 'keyword' to language specified by 'set_translation_language' using cloud service 'api'. You need to set API settings first.

Usage:

`Translator$automatic_translate(keyword, api = "google")`

Arguments:

`keyword` character or vector of characters with a word or expression to translate

`api` character with the name of the API you want to use. Currently supported: `google`.

Method `at()`**:** Wrapper to `automatic_translate` method

Usage:

`Translator$at(keyword, api = "google")`

Arguments:

`keyword` character or vector of characters with a word or expression to translate

`api` character with the name of the API you want to use. Currently supported: `google`.

Method `use_js()`**:** Call to wrap translation in span object. Used for browser-side translations.

Usage:

`Translator$use_js()`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

`Translator$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
  i18n <- Translator$new(translation_json_path = "translation.json") # translation file
  i18n$set_translation_language("it")
  i18n$t("This text will be translated to Italian")

## End(Not run)
```

```
# Shiny example
if (interactive()) {
library(shiny)
library(shiny.i18n)
 #to run this example  make sure that you have a translation file
 #in the same path
i18n <- Translator$new(translation_json_path = "examples/data/translation.json")
i18n$set_translation_language("pl")
ui <- fluidPage(
  h2(i18n$t("Hello Shiny!"))
)
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
}
```

`update_lang` *Update language (i18n) in UI*

Description

It sends a message to session object to update the language in UI elements.

Usage

```
update_lang(language, session = shiny::getDefaultReactiveDomain())
```

Arguments

`language`	character with language code
`session`	Shiny server session (default: current reactive domain)

See Also

usei18n

`usei18n` *Use i18n in UI*

Description

This is an auxiliary function needed to monitor the state of the UI for live language translations.

Usage

```
usei18n(translator)
```

Arguments

`translator`	shiny.i18 Translator object

Examples

```
if (interactive()) {
  library(shiny)
  library(shiny.i18n)
  # for this example to run make sure that you have a translation file
  # in the same path
  i18n <- Translator$new(translation_json_path = "translation.json")
  i18n$set_translation_language("en")

  ui <- fluidPage(
    usei18n(i18n),
    actionButton("go", "GO!"),
    h2(i18n$t("Hello Shiny!"))
  )

  server <- shinyServer(function(input, output, session) {
    observeEvent(input$go,{
      update_lang(session, "pl")
    })
  })

  shinyApp(ui = ui, server = server)
}
```

Index