Package: translated (via r-universe)

September 13, 2024

Type Package

Title Simple and Robust Translation System

Version 0.1.1

Date 2023-04-25

Description Allows translating with formatted string literals, grouped entries, and configurable system of plurals. Have a separate file for each locale and use inheritance to handle dialect differences.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports glue, jsonlite

Suggests covr, deepdep, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/ttscience/translated

BugReports https://github.com/ttscience/translated/issues

Repository https://ttscience.r-universe.dev

RemoteUrl https://github.com/ttscience/translated

RemoteRef HEAD

RemoteSha 4482de27ba8e5cf60652f2abf581a765df332505

Contents

trans *Translate strings*

Description

This function finds a translation for given keys in translation JSON files. If none found, a bare key is returned and a warning raised. Translation language is controlled by `trans_locale()` function.

Translation strings may contain `{var}` chunks. These are replaced in the final string by respective strings passed to `...`.

If a translation depends on a count, a `.n` parameter allows the user to select an appropriate translation form. Count interpretation is defined within translation JSONs for each language separately under `$.config.plural` path. Integer returned by this interpretation function is used as index for extracting the correct translation from an array.

Thanks to the implementation strategy, it is possible to create recursive translations (i.e. translations whose smaller chunks are translations themselves). This makes it possible to translate statements like "Found n file(s) in m directory/ies" by translating "files" and "directories" chunks first. In this case an appropriate translation entry would look like: "Found trans("file", .n = n_files) in trans("directory", .n = n_dirs)". Braces evaluate the content inside as R code.

Usage

```
trans(.key, ..., .n = NULL)
```

Arguments

`.key`	`character()` Keys to the translation map (accepts vectors of strings). Keys should not contain dots, unless it's a compound key (where dots separate parts of path to the key, e.g. `"submit_form.buttons.reset"`).
`...`	`character(1)` Named strings to substitute into translation; they replace substrings of the form `"{var}"` (`var` being a name of a string). Avoid names starting with dots to avoid incompatibility issues with future versions of this package.
`.n`	`integer(1)` Count for translating value-dependent messages, e.g. "n file(s)".

Value

A string vector of the same length as `.key`, each element being a translated string.

Examples

```
trans_path(system.file("examples", package = "translated"))

trans("title")
trans("btn_insert", number = "five")
```

```
# Non-character values are coerced to strings
trans("btn_insert", number = 5)

# Missing entries raise a warning and return the key
trans("missing_entry")
```

`trans_available` *Display available locales*

Description

This function displays available locales

Usage

```
trans_available()
```

Value

Returns available locales in df

Examples

```
path <- system.file("examples", package = "translated")
trans_path(path)

# Display available locales
trans_available()
```

`trans_locale` *Access translation locale*

Description

This function allows setting translation locale and accessing the currently set one.

Usage

```
trans_locale(locale)
```

Arguments

`locale`	`character(1)` Locale to set, must be of form `"lang_country.encoding"` (or simplified ones, i.e. `"lang_country"` and `"lang"`). `lang` and `country` must be two-letter codes, preferably in agreement with the latest ISO norm.

Value

If `locale` was not passed, currently set locale, else nothing.

Examples

```
trans_path(system.file("examples", package = "translated"))

# Check your default locale
trans_locale()

# Switch the translation to Polish language
trans_locale("pl_PL")
trans("title")
```

`trans_path` *Access translation path*

Description

This function allows setting translation path and accessing the currently set one.

Usage

```
trans_path(path)
```

Arguments

`path`	`character(1)` Path to set, should be a folder containing JSON files (possibly in subdirectories).

Value

If `path` was not passed, currently set path, else nothing.

Examples

```
# Set path to example translations shipped with this package
trans_path(system.file("examples", package = "translated"))

# Check the current path
trans_path()
```

`trans_reload` *Force translation reload*

Description

This function forces the translations to be loaded from JSON files again.

Usage

```
trans_reload()
```

Value

No return value, called for its side effect.

Examples

```
trans_path(system.file("examples", package = "translated"))
trans_reload()
```

Index