Package ‘spsUtil’

May 5, 2021

Title 'systemPipeShiny' Utility Functions

Version 0.1.2

Date 2021-05-04

Description The systemPipeShiny (SPS) framework comes with many useful utility functions. However, installing the whole framework is heavy and takes some time. If you like only a few useful utility functions from SPS, install this package is enough.

Depends R (>= 4.0.0)

Imports httr, assertthat, stringr, glue, magrittr, crayon, utils

Suggests testthat

License GPL (>= 3)

Encoding UTF-8

BugReports `https://github.com/lz100/spsUtil/issues`

URL `https://github.com/lz100/spsUtil`

RoxygenNote 7.1.1

Config/testthat/edition 3

NeedsCompilation no

Author Le Zhang [aut, cre]

Maintainer Le Zhang <`lezhang100@gmail.com`>

Repository CRAN

Date/Publication 2021-05-05 15:20:13 UTC

R topics documented:

`checkNameSpace` *check namespace*

Description

Help you to check if you have certain packages and return missing package names

Usage

```
checkNameSpace(packages, quietly = FALSE, from = "CRAN")
```

Arguments

`packages`	vector of strings
`quietly`	bool, give you warning on fail?
`from`	string, where this package is from like, "CRAN", "GitHub", only for output message display purpose

Value

vector of strings, of missing package names, `character(0)` if no missing

Examples

```
checkNameSpace("ggplot2")
checkNameSpace("random_pkg")
checkNameSpace("random_pkg", quietly = TRUE)
```

`checkUrl` *check if an URL can be reached*

Description

check if a URL can be reached, return TRUE if yes and FALSE if cannot or with other status code

Usage

```
checkUrl(url, timeout = 5)
```

Arguments

`url`	string, the URL to request
`timeout`	seconds to wait before return FALSE

Value

`TRUE` if url is reachable, `FALSE` if not

Examples

```
checkUrl("https://google.com")
try(checkUrl("https://randomwebsite123.com", 1))
```

`emptyIsFalse` *Empty objects and FALSE will return FALSE*

Description

judge if an object is empty or FALSE, and return FALSE if it is

Usage

```
emptyIsFalse(x)
```

Arguments

`x`	any R object

Details

not working on S4 class objects.

Useful for if statement. Normal empty object in if will spawn error. Wrap the expression with `emptyIsFalse` can avoid this. See examples

Value

`NA`, `""`, `NULL`, `length(0)`, `nchar == 0` and `FALSE` will return `FALSE`, otherwise `TRUE`.

Examples

```
emptyIsFalse(NULL)
emptyIsFalse(NA)
emptyIsFalse("")
try(`if(NULL) "not empty" else "empty"`) # will generate error
if(emptyIsFalse(NULL)) "not empty" else "empty" # this will work
# similar for `NA`, `""`, `character(0)` and more
```

`msg` *pretty logging message*

Description

If

1. `use_color` = `TRUE` or
2. under SPS main package `use_crayonoption` is `TRUE`
3. In a console that supports colors Then the message will be colorful, other wise no color

"INFO" level spawns `message`, "WARNING" is `warning`, "ERROR" spawns `stop`, other levels use `cat`.

`spsinfo`, `spswarn`, `spserror` are higher level wrappers of `msg`. The only difference is they have SPS- prefix.

`spsinfo` has an additional arg `verbose`. This arg works similarly to all other `verbose` args in SPS:

1. if not specified, it follows the project option. If SPS option `verbose` is set to `TRUE`, message will be displayed; if `FALSE`, mute the message.
2. It can be be forced to `TRUE` and `FALSE`. `TRUE` will forcibly generate the msg, and `FALSE` will mute the message.

Usage

```
msg(
  msg,
  level = "INFO",
  .other_color = NULL,
  info_text = "INFO",
  warning_text = "WARNING",
  error_text = "ERROR",
  use_color = TRUE
)

spsinfo(msg, verbose = NULL)

spswarn(msg)

spserror(msg)
```

Arguments

`msg`	a character string of message or a vector of character strings, each item in the vector presents one line of words
`level`	typically, one of "INFO", "WARNING", "ERROR", not case sensitive. Other custom levels will work too.

`.other_color`	hex color code or named colors, when levels are not in "INFO", "WARNING", "ERROR", this value will be used
`info_text`	info level text prefix, use with "INFO" level
`warning_text`	warning level text prefix, use with "WARNING" level
`error_text`	error level text prefix, use with "ERROR" level
`use_color`	bool, default `TRUE`, to use color if supported?
`verbose`	bool, default get from sps project options, can be overwritten

Details

1. If `use_color` is `TRUE`, output message will forcibly use color if the console has color support, ignore SPS `use_crayon` option.
2. If `use_color` is `FALSE`, but you are using within SPS framework, the `use_crayon` option is set to `TRUE`, color will be used.
3. Otherwise message will be no color.

Value

see description and details

Examples

```
msg("this is info")
msg("this is warning", "warning")
try(msg("this is error", "error"))
msg("this is another level", "my level", "green")
spsinfo("some msg, verbose false", verbose = FALSE) # will not show up
spsinfo("some msg, verbose true", verbose = TRUE)
spswarn("sps warning")
try(spserror("sps error"))
```

`quiet` *Suppress cat, print, message and warning*

Description

Useful if you want to suppress cat, print, message and warning. You can choose what to mute. Default all four methods are muted.

Usage

```
quiet(x, print_cat = TRUE, message = TRUE, warning = TRUE)
```

Arguments

`x`	function or expression or value assignment expression
`print_cat`	bool, mute `print` and `cat`?
`message`	bool, mute `messages`?
`warning`	bool, mute `warnings`?

Value

If your original functions has a return, it will return in `invisible(x)`

Examples

```
quiet(warning(123))
quiet(message(123))
quiet(print(123))
quiet(cat(123))
quiet(warning(123), warning = FALSE)
quiet(message(123), message = FALSE)
quiet(print(123), print_cat = FALSE)
quiet(cat(123), print_cat = FALSE)
```

`remove_ANSI` *Remove ANSI color code*

Description

Remove ANSI pre-/suffix-fix in a character string.

Usage

```
remove_ANSI(strings)
```

Arguments

`strings`	strings, a character vector

Value

strings with out ANSI characters

Examples

```
remove_ANSI("\033[34m\033[1ma\033[22m\033[39m")
remove_ANSI(c("\033[34m\033[1ma\033[22m\033[39m",
              "\033[34m\033[1mb\033[22m\033[39m"))
```

spsOption *Get or set SPS options*

Description

Get or set SPS options

Usage

```
spsOption(opt, value = NULL, empty_is_false = TRUE)
```

Arguments

`opt`	string, length 1, what option you want to get or set
`value`	if this is not `NULL`, this function will set the option you choose to this value
`empty_is_false`	bool, when trying to get an option value, if the option is `NULL`, `NA`, `""` or length is 0, return `FALSE`?

Value

return the option value if value exists; return `FALSE` if the value is empty, like `NULL`, `NA`, `""`; return `NULL` if `empty_is_false = FALSE`; see emptyIsFalse

If `value != NULL` will set the option to this new value, no returns.

Examples

```
spsOption("test1") # get a not existing option
spsOption("test1", 1) # set the value
spsOption("test1") # get the value again
spsOption("test2")
spsOption("test2", empty_is_false = FALSE)
```

Index