Package 'DIZutils'

June 3, 2020

Title Utilities for 'DIZ' R Package Development

Version 0.0.4

Date 2020-05-29

Description Utility functions used for R package development infrastructure inside the data integration centers ('DIZ') to standardize and facilitate repetitive tasks such as setting up a database connection or issuing notification messages and to avoid redundancy.

License GPL-3

URL https://gitlab.miracum.org/miracum/dqa/dizutils

BugReports https://gitlab.miracum.org/miracum/dqa/dizutils/issues

Depends R (>= 2.10)

Imports config, data.table, DBI, RJDBC, RPostgres, shiny, shinyjs

Suggests knitr, lintr, qpdf, rmarkdown, testthat

biocViews

Copyright Universitätsklinikum Erlangen

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author Jonathan M. Mang [aut, cre] (<https://orcid.org/0000-0003-0518-4710>), Lorenz A. Kapsner [aut] (<https://orcid.org/0000-0003-1866-860X>), MIRACUM - Medical Informatics in Research and Care in University Medicine [fnd]

Maintainer Jonathan M. Mang <jonathan.mang@uk-erlangen.de>

Repository CRAN

Date/Publication 2020-06-03 16:30:12 UTC

R topics documented:

`cleanup_old_logfile` *Archives the current logfile and creates a new blank one.*

Description

This function is called once at the beginning of the runtime of the tool. It checks whether there is an old logfile and renames it (if existing) to "logfile_20xx-xx-xx-xxxxxx.log". Then a new, empty, logfile "logfile.log" is created.

Usage

```
cleanup_old_logfile(logfile_dir)
```

Arguments

`logfile_dir`	(Optional, String, default: "tempdir()") The absolute path to folder where the logfile will be stored.

Value

No return value, called for side effects (see description)

Examples

```
## Not run:
cleanup_old_logfile("path/to/logfile/dir/")

## End(Not run)
```

`clean_path_name` *clean_path_name helper function*

Description

Function to clean paths to have a tailing slash

Usage

```
clean_path_name(pathname)
```

Arguments

`pathname`	A character string. A path name to be cleaned (to have a tailing slash).

Value

The result is the input but with an tailing slash.

Examples

```
## Not run:
# Both function calls will return "home/test/"
clean_path_name("home/test")
clean_path_name("home/test/")

## End(Not run)
```

`close_all_connections` *Cleanup function to unset/close all open connections*

Description

This function is meant to be called at the end of a run of the app. It will close all open connections to files.

Usage

```
close_all_connections(logfile_dir, headless)
```

Arguments

logfile_dir	(Optional, String, default: "tempdir()") The absolute path to folder where the logfile will be stored.
headless	(Optional, Boolean, default: TRUE) Indicating, if the function is run only in the console (headless = TRUE) or on a GUI frontend (headless = FALSE).

Value

No return value, called for side effects (see description)

Examples

```
## Not run:
close_all_connections("path/to/logfile/dir/", TRUE)

## End(Not run)
```

---

db_connection *db_connection helper function*

Description

Internal function to test and get the database connection of the target data system.

Usage

```
db_connection(
  db_name,
  db_type,
  headless = FALSE,
  from_env = TRUE,
  settings = NULL,
  timeout = 30,
  logfile_dir = NULL,
  lib_path = NULL
)
```

Arguments

db_name	A character. Name of the database system.
db_type	A character. Type of the database system. Currently implemented systems are: 'postgres', 'oracle'.
headless	A boolean (default: FALSE). Indicating, if the function is run only in the console (headless = TRUE) or on a GUI frontend (headless = FALSE).
from_env	A boolean (default: TRUE). Should database connection be read from the environment or from a settings file.

settings	A list. Required if 'from_env=TRUE'. A list containing settings for the database connection. Required fields are 'host', 'db_name', 'port', 'user' and 'password'. Additionally for Oracle DB's: 'sid'.
timeout	A timeout in sec. for the db-connection establishment. Values below 2 seconds are not recommended. Default is 30 seconds.
logfile_dir	(Optional, String, default: "tempdir()") The absolute path to folder where the logfile will be stored.
lib_path	A character string. The path to the ojdbc7.jar file.

Value

If successful, the result will be the established connection. Otherwise the result will be null.

Examples

```
## Not run:
db_con <- DIZutils::db_connection(
  db_name = "i2b2",
  db_type = "postgres",
  headless = true,
  logfile_dir = "./path/to/logfile/dir/"
)
## End(Not run)
```

---

feedback *Function to feedback messages either to the user and/or to the console and to the logfile.*

Description

This function provides the functionality to publish any kind of information to the user, the console and/or to the logfile. This might be a simple info, a warning or an error. The function can be used to select the output (console, ui, logfile). If no output is selected, the print_this string will be printed to the console and to logfile. One of these must be a string with length > 0: print_me, console, ui

Usage

```
feedback(
  print_this = "",
  type = "Info",
  ui = FALSE,
  console = TRUE,
  logfile = TRUE,
  logjs = FALSE,
  prefix = "",
  suffix = "",
```

```
  findme = "",
  logfile_dir = tempdir(),
  headless = TRUE
)
```

Arguments

print_this	(Optional, String, default: "")
type	(Optional, String, default: "Info") E.g. "Warning", "Error. Default: "Info"
ui	(Optional, Boolean/String, default: FALSE) If true, the message will also be printed to the user in form of a modal. Can also be a string.
console	(Optional, Boolean/String, default: TRUE) If true, the message will also be printed to the console as is. Can also be a string.
logfile	(Optional, Boolean, default: TRUE) If true (default) the print_this string will also be printed to the console.
logjs	(Optional, Boolean, default: FALSE) If true (default: false) the print_this string will also be printed to the javascript-console. This only makes sense, if the gui is active.
prefix	Prefix (Optional, String, default: "") This is useful if print_this is an array/list. Each entry will then be new row with this prefix.
suffix	Suffix (Optional, String, default: "") Same like prefix but at the end of each line.
findme	(Optional, String, default: "") Recommended with length 10. String to find the message in the code. E.g. 10-digit random hex from (<https://www.browserling.com/tools/random-hex>) or (<https://onlinerandomtools.com/generate-random-hexadecimal-numbers>)
logfile_dir	(Optional, String, default: "tempdir()") The absolute path to folder where the logfile will be stored.
headless	(Optional, Boolean, default: TRUE) Indicating, if the function is run only in the console (headless = TRUE) or on a GUI frontend (headless = FALSE).

Value

No return value, called for publishing a message.

Examples

```
## Not run:
feedback(
  print_this = "Error occured when counting source_data",
  type = "Error",
  findme = "255bb3695c",
  logfile_dir = rv$logfile_dir,
  headless = rv$headless
)
## End(Not run)
```

`feedback_get_formatted_string` *Format the feedback string*

Description

Helper function for the feedback function to combine the input parameters in proper manner to ge a pretty and informative string which than can be added to the logfile and/or be displayed in the console. CAUTION: 'print_this' must be of length 1! For arrays loop through them by hand and call this function several times! Internal use. Use the robust 'feedback' function instead.

Usage

```
feedback_get_formatted_string(print_this, type, findme, prefix, suffix)
```

Arguments

`print_this`	(Optional, String, default: "")
`type`	(Optional, String, default: "Info") E.g. "Warning", "Error. Default: "Info"
`findme`	(Optional, String, default: "") Recommended with length 10. String to find the message in the code. E.g. 10-digit random hex from (<https://www.browserling.com/tools/random-hex>) or (<https://onlinerandomtools.com/generate-random-hexadecimal-numbers>)
`prefix`	Prefix (Optional, String, default: "") This is useful if print_this is an array/list. Each entry will then be new row with this prefix.
`suffix`	Suffix (Optional, String, default: "") Same like prefix but at the end of each line.

Value

Returns a properly an consistent formatted string containing the parameters handed over to this function.

`feedback_to_console` *Print to the console. Internal use.*

Description

Helper function for the feedback function to print stuff to the console. Everything will also be added to the logfile. Internal use. Use the robust 'feedback' function instead.

Usage

```
feedback_to_console(
  print_this,
  type,
  findme,
  prefix,
  suffix,
  logjs,
  logfile_dir
)
```

Arguments

print_this	(Optional, String, default: "")
type	(Optional, String, default: "Info") E.g. "Warning", "Error. Default: "Info"
findme	(Optional, String, default: "") Recommended with length 10. String to find the message in the code. E.g. 10-digit random hex from (<https://www.browserling.com/tools/random-hex>) or (<https://onlinerandomtools.com/generate-random-hexadecimal-numbers>)
prefix	Prefix (Optional, String, default: "") This is useful if print_this is an array/list. Each entry will then be new row with this prefix.
suffix	Suffix (Optional, String, default: "") Same like prefix but at the end of each line.
logjs	(Optional, Boolean, default: FALSE) If true (default: false) the print_this string will also be printed to the javascript-console. This only makes sense, if the gui is active.
logfile_dir	(Optional, String, default: "tempdir()") The absolute path to folder where the logfile will be stored.

Value

No return value, called for side effects (see description)

---

feedback_to_logfile *Add to the logfile. Internal use.*

Description

Helper function for the feedback function to add content to the logfile. Internal use. Use the robust 'feedback' function instead.

Usage

```
feedback_to_logfile(print_this, type, findme, prefix, suffix, logfile_dir)
```

Arguments

print_this	(Optional, String, default: "")
type	(Optional, String, default: "Info") E.g. "Warning", "Error. Default: "Info"
findme	(Optional, String, default: "") Recommended with length 10. String to find the message in the code. E.g. 10-digit random hex from (<https://www.browserling.com/tools/random-hex>) or (<https://onlinerandomtools.com/generate-random-hexadecimal-numbers>)
prefix	Prefix (Optional, String, default: "") This is useful if print_this is an array/list. Each entry will then be new row with this prefix.
suffix	Suffix (Optional, String, default: "") Same like prefix but at the end of each line.
logfile_dir	(Optional, String, default: "tempdir()") The absolute path to folder where the logfile will be stored.

Value

No return value, called for side effects (see description)

---

feedback_to_logjs *Feedback to the gui/browser-console with logjs. Internal use.*

Description

Helper function for the feedback function to also show the messages to the gui/user via the browser console. Internal use. Use the robust 'feedback' function instead.

Usage

```
feedback_to_logjs(print_this, logfile_dir, headless)
```

Arguments

print_this	(Optional, String, default: "")
logfile_dir	(Optional, String, default: "tempdir()") The absolute path to folder where the logfile will be stored.
headless	(Optional, Boolean, default: TRUE) Indicating, if the function is run only in the console (headless = TRUE) or on a GUI frontend (headless = FALSE).

Value

No return value, called for side effects (see description)

feedback_to_ui *Feedback to the user with a modal. Internal use.*

Description

Helper function for the feedback function to show modals to the gui/user. Everything will also be added to the logfile. Internal use. Use the robust 'feedback' function instead.

Usage

```
feedback_to_ui(print_this, type, logfile_dir, headless)
```

Arguments

print_this	(Optional, String, default: "")
type	(Optional, String, default: "Info") E.g. "Warning", "Error. Default: "Info"
logfile_dir	(Optional, String, default: "tempdir()") The absolute path to folder where the logfile will be stored.
headless	(Optional, Boolean, default: TRUE) Indicating, if the function is run only in the console (headless = TRUE) or on a GUI frontend (headless = FALSE).

Value

No return value, called for side effects (see description)

firstup *Converts the first letter of the input string to uppercase*

Description

Converts the first letter of the input string to uppercase

Usage

```
firstup(x)
```

Arguments

x	A character string. E.g. "hello world" will become "Hello world".

Value

Returns the input string but with a capital first letter.

Examples

```
## Not run:
firstup("first letter will be upper case as return")

## End(Not run)
```

`get_config` *get_config helper function*

Description

Internal function to read config files

Usage

```
get_config(config_file, config_key, logfile_dir, headless)
```

Arguments

`config_file`	A character string. The path to the config.yml-file containing the database configuration.
`config_key`	A character string. The name of the corresponding database. This string must be conform with the corresponding config section in the config.yml-file.
`logfile_dir`	(Optional, String, default: "tempdir()") The absolute path to folder where the logfile will be stored.
`headless`	(Optional, Boolean, default: TRUE) Indicating, if the function is run only in the console (headless = TRUE) or on a GUI frontend (headless = FALSE).

Value

If successful it returns the config, Null otherwise.

Examples

```
## Not run:
config <- get_config(
  config_file = paste0(utils_path, "/MISC/email.yml"),
  config_key = "email",
  logfile_dir = rv$logfile_dir,
  headless = rv$headless
)
## End(Not run)
```

get_config_env *get_config_env helper function*

Description

Internal function to read settings for a certain system from the environment

Usage

```
get_config_env(system_name, logfile_dir, headless)
```

Arguments

system_name	The name of the system (This is also the prefix used to get the environment variables with 'SYSTEM_KEY', e.g. 'I2B2_DBNAME')
logfile_dir	(Optional, String, default: "tempdir()") The absolute path to folder where the logfile will be stored.
headless	A boolean (default: FALSE). Indicating, if the function is run only in the console (headless = TRUE) or on a GUI frontend (headless = FALSE).

Value

If successful it returns the config, null otherwise.

Examples

```
## Not run:
get_config_env(
  system_name = "i2b2",
  logfile_dir = "/path/to/logfile/",
  headless = FALSE
)
## End(Not run)
```

global_env_hack *global_env_hack*

Description

Hack variable into global env (bypasses R CMD checks).

Usage

```
global_env_hack(key, val, pos = 1)
```

Arguments

key	A character (!) string. The name of the assigned variable
val	An object. The object that will be assigned to 'key'.
pos	An integer. The position of the environment (default: 1).

Value

No return value, called for side effects (see description).

See Also

http://adv-r.had.co.nz/Environments.html

Examples

```
## Not run:
global_env_hack(
  key = "utils_path",
  val = utils_path,
  pos = 1L
)
## End(Not run)
```

---

query_database *query_database helper function*

Description

Internal function to query the database. The function sends a sql statement to the database and returns a data.table.

Usage

```
query_database(db_con, sql_statement)
```

Arguments

db_con	A DBI database connection.
sql_statement	A character string containing a valid SQL statement.

Value

Returns the result of the db-query.

Examples

```
## Not run:
query_database(
  db_con = db_con,
  sql_statement = "SELECT * FROM table_name;"
)
## End(Not run)
```

`set_env_vars` *set_env_vars helper function*

Description

Internal function to set environment variables that are necessary for the database connections with `db_connection`.

Usage

```
set_env_vars(env_file)
```

Arguments

`env_file`	A character. The full path including the file name to the file containing the environment variable defintions to be loaded.

Value

No return value, called for side effects (see description)

See Also

`Sys.setenv`

Examples

```
## Not run:
set_env_vars("./.env")

## End(Not run)
```

`%notin%` *notin helper function*

Description

Function to return elements of x that are not in y.

Usage

```
x %notin% y
```

Arguments

x	Object 1.
y	Object 2.

Value

Returns the result of !

Examples

```
## Not run:
tmp1 <- c("a","b","c")
tmp2 <- c("b", "c", "d")
tmp1 %notin% tmp2

## End(Not run)
```

Index