

Package ‘DIZutils’

November 17, 2020

Title Utilities for 'DIZ' R Package Development

Version 0.0.6

Date 2020-11-17

Description Utility functions used for the 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://github.com/miracum/misc-dizutils>

BugReports <https://github.com/miracum/misc-dizutils/issues>

Depends R (>= 3.1.0)

Imports config, data.table, DBI (>= 1.1.0), RJDBC, RPostgres, shiny, shinyjs

Suggests lintr, testthat

Encoding UTF-8

Language en-US

LazyData true

SystemRequirements libpq >= 9.0: libpq-dev (deb) or postgresql-devel (rpm)

RoxygenNote 7.1.1

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], Universitätsklinikum Erlangen [cph]

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

Repository CRAN

Date/Publication 2020-11-17 13:10:04 UTC

R topics documented:

cleanup_old_logfile	2
clean_path_name	3
close_all_connections	3
db_connection	4
feedback	5
feedback_get_formatted_string	7
feedback_to_console	7
feedback_to_logfile	8
feedback_to_logjs	9
feedback_to_ui	10
firstup	10
get_config	11
get_config_env	12
global_env_hack	12
query_database	13
set_env_vars	14
%notin%	15

Index	16
--------------	-----------

cleanup_old_logfile	<i>Archives the current logfile and creates a new blank one.</i>
---------------------	--

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

```
cleanup_old_logfile("path/to/logfile/dir/")
```

clean_path_name	<i>clean_path_name helper function</i>
-----------------	--

Description

Function to clean paths to have a trailing slash

Usage

```
clean_path_name(pathname)
```

Arguments

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

Value

The result is the input but with an trailing slash.

Examples

```
# Both function calls will return "home/test/"
clean_path_name("home/test")
clean_path_name("home/test/")
```

close_all_connections	<i>Cleanup function to unset/close all open connections</i>
-----------------------	---

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

```
close_all_connections(
  logfile_dir = tempdir(),
  headless = TRUE
)
```

<code>db_connection</code>	<i>db_connection helper function</i>
----------------------------	--------------------------------------

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

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

Value

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

See Also

[dbConnect](#), [RPostgres](#)

Examples

```
## Not run:
db_con <- DIZutils::db_connection(
  db_name = "i2b2",
  db_type = "postgres",
  headless = TRUE,
  logfile_dir = tempdir()
)
## 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

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

Value

No return value, called for publishing a message.

Examples

```
feedback(  
  print_this = "Error occured when counting source_data",  
  type = "Error",  
  findme = "255bb3695c",  
  logfile_dir = tempdir(),  
  headless = TRUE  
)
```

`feedback_get_formatted_string`*Format the feedback string*

Description

Helper function for the feedback function to combine the input parameters in proper manner to get a pretty and informative string which 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

<code>print_this</code>	(Optional, String, default: "")
<code>type</code>	(Optional, String, default: "Info") E.g. "Warning", "Error. Default: "Info"
<code>findme</code>	(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 >)
<code>prefix</code>	Prefix (Optional, String, default: "") This is useful if <code>print_this</code> is an array/list. Each entry will then be new row with this prefix.
<code>suffix</code>	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

```
firstup("first letter of this string will be upper case as return")
```

get_config	<i>get_config helper function</i>
------------	-----------------------------------

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

```
utils_path <- tempdir()
config <- get_config(
  config_file = paste0(utils_path, "/MISC/email.yml"),
  config_key = "email",
  logfile_dir = tempdir(),
  headless = TRUE
)
```

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

```
get_config_env(
  system_name = "i2b2",
  logfile_dir = tempdir(),
  headless = FALSE
)
```

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

```
utils_path <- tempdir()
global_env_hack(
  key = "utils_path",
  val = utils_path,
  pos = 1L
)
```

query_database	<i>query_database helper function</i>
----------------	---------------------------------------

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:
db_con <- DIZutils::db_connection(
  db_name = "i2b2",
  db_type = "postgres"
)

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 definitions to be loaded.

Value

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

See Also

Sys.setenv

Examples

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

%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

```
tmp1 <- c("a","b","c")  
tmp2 <- c("b", "c", "d")  
tmp1 %notin% tmp2
```

Index

`%notin%`, [15](#)

`clean_path_name`, [3](#)

`cleanup_old_logfile`, [2](#)

`close_all_connections`, [3](#)

`db_connection`, [4](#)

`dbConnect`, [5](#)

`feedback`, [5](#)

`feedback_get_formatted_string`, [7](#)

`feedback_to_console`, [7](#)

`feedback_to_logfile`, [8](#)

`feedback_to_logjs`, [9](#)

`feedback_to_ui`, [10](#)

`firstup`, [10](#)

`get_config`, [11](#)

`get_config_env`, [12](#)

`global_env_hack`, [12](#)

`query_database`, [13](#)

`RPostgres`, [5](#)

`set_env_vars`, [14](#)