

Package: configulaR (via r-universe)

May 27, 2026

Title Manage Application Settings via '.env' or '.ini' Files

Version 0.1.0

Description Provides a simple way to manage application settings by loading configuration values from '.env' or '.ini' files. It supports default values, type casting, and environment variable overrides, enabling a clean separation of configuration from code. Ideal for managing credentials, API keys, and deployment-specific settings.

License MIT + file LICENSE

URL <https://dataupsurge.github.io/configulaR/>,
<https://github.com/dataupsurge/configulaR>

BugReports <https://github.com/dataupsurge/configulaR/issues>

Imports assertthat, here, magrittr, stringr

Suggests covr (>= 3.6.4), docopt (>= 0.7.1), git2r (>= 0.35.0), linter (>= 3.1.2), precommit (>= 0.4.3), roxygen2 (>= 7.3.2), styler (>= 1.10.3), testthat (>= 3.0.0), withr (>= 2.1.2)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Config/pak/sysreqs libicu-dev

Repository <https://dataupsurge.r-universe.dev>

Date/Publication 2025-04-18 08:49:39 UTC

RemoteUrl <https://github.com/dataupsurge/configular>

RemoteRef HEAD

RemoteSha b797935e57de779da7fb97d5e5189c958f77cc65

Contents

cast_as	2
find_config_files	3
get_config	3
get_config_filename_from_dir	4
get_var	5
is_undefined	6
load_dot_env	6
parse_dot_line	7
random_string	8
remove_comments	8
remove_empty_lines	9
to_logical	9

Index	10
--------------	-----------

cast_as	<i>Helper function to cast data from keyword of function</i>
---------	--

Description

Helper function to cast data from keyword of function

Usage

```
cast_as(variable, cast)
```

Arguments

variable	String to be casted
cast	Casting function or keyword

Value

casted variable

find_config_files	<i>Config file search</i>
-------------------	---------------------------

Description

Config file search

Usage

```
find_config_files(path)
```

Arguments

path	The directory from which the config file search is initialized. Look in current and root directories until finding a config file reaching the primary root.
------	---

Value

the full path of the config file found. If no config file are found, it returns a 'NULL'.

get_config	<i>Search for config files and return variable / value pairs as a list.</i>
------------	---

Description

configulaR currently supports only .env files. 'get_config' will search if the specified directory and its parent a '.env' file. If no file is found, a empty list is returned.

Usage

```
get_config(path = NULL)
```

Arguments

path	a path from where a config if searched for. Parent directories will be evaluated if no config file is found is the specified directory. If NULL, the current directory will be taken as default.
------	--

Details

Env file:

Simply create a .env text file on your repository's root directory in the form:

```
DEBUG=True
TEMPLATE_DEBUG=True
SECRET_KEY=ARANDOMSECRETKEY
DATABASE_URL=mysql://myuser:mypassword@myhost/mydatabase
PERCENTILE=90
#COMMENTED=42
```

Value

a list with variables / values from config file. If no config file has been found, an empty list is returned.

Examples

```
config <- get_config()
config
```

```
get_config_filename_from_dir
      Return config file path found in a directory
```

Description

Return config file path found in a directory

Usage

```
get_config_filename_from_dir(path)
```

Arguments

path The directory were config files are looking for.

Value

the full path of the config file found. If no config file are found, it returns a 'NULL'.

get_var	<i>Retrieve the value of the variable</i>
---------	---

Description

Get a variable from, in order of priority, environment variable, .env or settings.ini, or default values. Data type can be cast to boolean or integer.

Usage

```
get_var(  
  config = NULL,  
  var_name = NULL,  
  path = NULL,  
  default = structure("UNDEFINED_", class = "UNDEFINED_"),  
  cast = NULL  
)
```

Arguments

config	an object returned by the <code>init_config</code> function. If <code>NULL</code> , the config will be evaluated from the <code>path</code> argument.
var_name	the variable of interest
path	the path from where config files are searched for. If <code>NULL</code> , the current directory will be considered as default.
default	a default value
cast	Either a function or a type of cast. Currently implemented options are 'integer', 'boolean' or 'float'.

Value

The value associated to the config variable. The type depends on the `cast` argument. Default is string.

Examples

```
config <- get_config()  
get_var(config, "test", default = "yes", cast = "bool")
```

is_undefined	<i>Test if object if of class _UNDEFINED</i>
--------------	--

Description

Test if object if of class _UNDEFINED

Usage

```
is_undefined(x)
```

Arguments

x an object.

Value

a boolean

load_dot_env	<i>Load environment variables from the specified file</i>
--------------	---

Description

The file is parsed, and line is expected to have one of the following formats:

```
VARIABLE=value  
VARIABLE2="quoted value"  
VARIABLE3='another quoted variable'  
# Comment line  
export EXPORTED="exported variable"  
export EXPORTED2=another
```

Usage

```
load_dot_env(fpath = ".env")
```

Arguments

fpath The path to the ‘.env’ config file

Details

Load variables defined in the given file, as environment variables.

Detailed specification:

- A leading `export` is ignored, to keep the file compatible with Unix shells.
- No whitespace is allowed right before or after the equal sign, again, to promote compatibility with Unix shells.
- No multi-line variables are supported currently. The file is strictly parsed line by line.
- Unlike for Unix shells, unquoted values are *not* terminated by whitespace.
- Comments start with `#`, without any leading whitespace. You cannot mix variable definitions and comments in the same line.
- Empty lines (containing whitespace only) are ignored.

It is suggested to keep the file in a form that is parsed the same way with `dotenv` and `bash` (or other shells).

Value

A named list of environment variables, where names are the variable names and values are the variable values

Examples

```
# Load from a file
tmp <- tempfile()

cat("dotenvexamplefoo=bar\n", file = tmp)
load_dot_env(tmp)

# Clean up
unlink(tmp)
```

parse_dot_line	<i>Parse environment variables from lines in a .env file</i>
----------------	--

Description

Parse environment variables from lines in a `.env` file

Usage

```
parse_dot_line(lines)
```

Arguments

`lines` Character vector containing lines from a `.env` file

Value

A named list where names are environment variable names and values are their corresponding values

random_string	<i>Generate a random string of specified length</i>
---------------	---

Description

Generate a random string of specified length

Usage

```
random_string(string_length = 6, replace = TRUE)
```

Arguments

string_length	Integer, length of the string to generate.
replace	Boolean, Use replace in the sampling procedure.

Value

A random string.

remove_comments	<i>Parse config file and remove comment lines</i>
-----------------	---

Description

Parse config file and remove comment lines

Usage

```
remove_comments(lines)
```

Arguments

lines	A list of config file lines.
-------	------------------------------

Value

A list of config file lines where comment lines have been filtered out.

remove_empty_lines	<i>Parse config file and remove empty lines</i>
--------------------	---

Description

Parse config file and remove empty lines

Usage

```
remove_empty_lines(lines)
```

Arguments

lines	A list of config file lines.
-------	------------------------------

Value

A list of config file lines where empty lines have been filtered out.

to_logical	<i>Convert categorical representations of true/false values to a logical</i>
------------	--

Description

Allow the conversion of logical related data to actual Boolean

Usage

```
to_logical(x)
```

Arguments

x	a vector of boolean compatible values.
---	--

Value

a vector of boolean

Examples

```
to_logical(c("yes", "no"))
```

Index

`cast_as`, 2

`find_config_files`, 3

`get_config`, 3

`get_config_filename_from_dir`, 4

`get_var`, 5

`is_undefined`, 6

`load_dot_env`, 6

`parse_dot_line`, 7

`random_string`, 8

`remove_comments`, 8

`remove_empty_lines`, 9

`to_logical`, 9