Package: config (via r-universe)

September 27, 2024

Type Package

Title Manage Environment Specific Configuration Values

Version 0.3.2.9000

Imports yaml (>= 2.1.19)

Suggests testthat, knitr, rmarkdown, covr, spelling, jsonlite, withr

Description Manage configuration values across multiple environments (e.g. development, test, production). Read values using a function that determines the current environment and returns the appropriate value.

License GPL-3

URL https://rstudio.github.io/config/, https://github.com/rstudio/config

BugReports https://github.com/rstudio/config/issues

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

Config/testthat/edition 3

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

RemoteUrl https://github.com/rstudio/config

RemoteRef HEAD

RemoteSha 09d687ebddc05bb58b7919d66b3101f46cd51421

Contents

`get` *Read configuration values. Always use as* `config::get()`.

Description

Read from the currently active configuration, retrieving either a single named value or all values as a list.

Usage

```
get(
  value = NULL,
  config = Sys.getenv("R_CONFIG_ACTIVE", "default"),
  file = Sys.getenv("R_CONFIG_FILE", "config.yml"),
  use_parent = TRUE
)
```

Arguments

`value`	Name of value (`NULL` to read all values)
`config`	Name of configuration to read from. Defaults to the value of the `R_CONFIG_ACTIVE` environment variable ("default" if the variable does not exist).
`file`	Configuration file to read from (defaults to `"config.yml"`). If the file isn't found at the location specified then parent directories are searched for a file of the same name.
`use_parent`	`TRUE` to scan parent directories for configuration files if the specified config file isn't found.

Details

For additional details see https://rstudio.github.io/config/.

Value

The requested configuration value (or all values as a list of `NULL` is passed for `value`).

A list, or vector, corresponding to the contents of the config file.

Warning - Do not attach the package using library(config)

We strongly recommend you use `config::get()` rather than attaching the package using `library(config)`.

In fact, we strongly recommend you never use `library(config)`.

The underlying reason is that the `get()` and `merge()` functions in `{config}` will mask these functions with the same names in base R.

See Also

is_active(), merge()

Examples

```
yaml <- "
default:
  trials: 5
  dataset: 'data-sampled.csv'

production:
  trials: 30
  dataset: 'data.csv'
"

get <- base::get



with_config(yaml, config::get())
with_config(yaml, config::get("trials"))
```

---

is_active *Test active configuration.*

Description

Check whether a configuration is currently active.

Usage

```
is_active(config)
```

Arguments

config	Configuration name

Details

The name of the currently active configuration is read from the R_CONFIG_ACTIVE environment variable. If the variable is not defined then the "default" configuration is used.

To test for whether a configuration is active you should use the is_active() function rather than inspecting the environment variable directly (this is to so that tests remain valid if other means of specifying configurations are introduced in the future).

Value

Logical indicating whether the specified configuration is active

See Also

get()

---

merge *Merge two configurations. Always use as* config::merge().

Description

Merge one configuration into another recursively.

Usage

```
merge(base_config, merge_config)
```

Arguments

base_config	Configuration to merge values into
merge_config	Configuration to merge values from

Value

Configuration which includes the values from merge_config merged into base_config.

Warning - Do not attach the package using library(config)

We strongly recommend you use config::get() rather than attaching the package using library(config).

In fact, we strongly recommend you never use library(config).

The underlying reason is that the get() and merge() functions in {config} will mask these functions with the same names in base R.

See Also

get()

`with_config` *Run code using a temporary config file.*

Description

This function takes inspiration from `withr::with_envvar()` and may be useful for testing purposes.

Usage

```
with_config(
  config_yml,
  code,
  .active_config = c(R_CONFIG_ACTIVE = "default"),
  .extra_env_vars = NULL
)
```

Arguments

`config_yml`	Either the path to a config file, or a character string representing a yaml configuration.
`code`	Code to execute in a temporary environment.
`.active_config`	Either a string representing a configuration, e.g. `default`, or a named character representing an environment variable, e.g. `c(R_CONFIG_ACTIVE = "default")`.
`.extra_env_vars`	Additional environment variables to set.

Value

The result of running the `code`, after having temporarily set the necessary environment variables.

Examples

```
yaml <- '
default:
  db_name: dbase
  databases:
    db1: !expr paste0(db_name, "/one")
    db2: !expr paste0(db_name, "/two")

staging:
  staging_postfix: _staging
  db_name: dbase
  databases:
    db1: !expr paste0(db_name, staging_postfix, "/one")
    db2: !expr paste0(db_name, staging_postfix, "/two")
'
```

```
# Ensure that base::get() doesn't get masked, for tests on CRAN
get <- base::get

with_config(yaml, config::get() )
with_config(yaml, config::get("databases", config = "default") )
with_config(yaml, config::get("databases", config = "staging") )

config_file <- system.file("tests/testthat/config.yml", package = "config")
if (file.exists(config_file)) {
  with_config(config_file, config::get())
}
```

Index