Package: envsetup (via r-universe)

October 17, 2024

Title Support the Setup of the R Environment for Clinical Trial Programming Workflows

Version 0.2.0

Description The purpose of this package is to support the setup the R environment. The two main features are 'autos', to automatically source files and/or directories into your environment, and 'paths' to consistently set path objects across projects for input and output. Both are implemented using a configuration file to allow easy, custom configurations that can be used for multiple or all projects.

License Apache License 2.0

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports config, fs, purrr, rlang, usethis

Suggests rmarkdown, testthat, knitr, kableExtra, magrittr, devtools, readr, tidyr, withr, lintr, styler, renv, covr

VignetteBuilder knitr

Config/testthat/edition 3

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

RemoteUrl https://github.com/pharmaverse/envsetup

RemoteRef HEAD

RemoteSha 6bc700344ad870a44444aaa2862bda59b223c216

Contents

`build_from_config` *Build directory structure from a configuration file*

Description

Build directory structure from a configuration file

Usage

```
build_from_config(config, root = NULL)
```

Arguments

`config`	configuration object from config::get() containing paths#'
`root`	root directory to build from. Leave as NULL if using absolute paths. Set to working directory if using relative paths.

Value

Called for its side-effects. The directories build print as a tree-like format from `fs::dir_tree()`.

Examples

```
tmpdir <- tempdir()

hierarchy <- "default:
  paths:
    data: !expr list(DEV = '/demo/DEV/username/project1/data',
                     PROD = '/demo/PROD/project1/data')
    output: !expr list(DEV = '/demo/DEV/username/project1/output',
                       PROD = '/demo/PROD/project1/output')
    programs: !expr list(DEV = '/demo/DEV/username/project1/programs',
                         PROD = '/demo/PROD/project1/programs')
    docs: !expr list(DEV = 'docs',
                     PROD = 'docs')"

writeLines(hierarchy, file.path(tmpdir, "hierarchy.yml"))

config <- config::get(file = file.path(tmpdir, "hierarchy.yml"))

build_from_config(config, tmpdir)
```

detach_autos *Detach the autos from the current session*

Description

This function will remove any autos that have been set from the search path

Usage

```
detach_autos()
```

Value

Called for its side-effects.

Examples

```
tmpdir <- tempdir()
print(tmpdir)

# account for windows
if (Sys.info()['sysname'] == "Windows") {
  tmpdir <- gsub("\\", "\\\\", tmpdir, fixed = TRUE)
}

# Create an example config file\
hierarchy <- paste0("default:
  paths:
    functions: !expr list(DEV = file.path('",tmpdir,"',
                                          'demo',
                                          'DEV',
                                          'username',
                                          'project1',
                                          'functions'),
                          PROD = file.path('",tmpdir,"',
                                           'demo',
                                           'PROD',
                                           'project1',
                                           'functions'))
  autos:
     my_functions: !expr list(DEV = file.path('",tmpdir,"',
                                              'demo',
                                              'DEV',
                                              'username',
                                              'project1',
                                              'functions'),
                              PROD = file.path('",tmpdir,"',
                                               'demo',
                                               'PROD',
                                               'project1',
```

```
                                                'functions'))")

# write config
writeLines(hierarchy, file.path(tmpdir, "hierarchy.yml"))

config <- config::get(file = file.path(tmpdir, "hierarchy.yml"))

build_from_config(config)

# write function to DEV
writeLines("dev_function <- function() {print(environment(dev_function))}",
       file.path(tmpdir, 'demo', 'DEV', 'username', 'project1', 'functions', 'dev_function.r'))

# write function to PROD
writeLines("prod_function <- function() {print(environment(prod_function))}",
          file.path(tmpdir, 'demo', 'PROD', 'project1', 'functions', 'prod_function.r'))

# setup the environment
Sys.setenv(ENVSETUP_ENVIRON = "DEV")
rprofile(config::get(file = file.path(tmpdir, "hierarchy.yml")))

# show dev_function() and prod_function() are available and print their location
dev_function()
prod_function()

# remove autos from search
detach_autos()
```

`init` *Initialize the R environment with envsetup*

Description

Initialize the R environment with envsetup

Usage

```
init(project, config_path = NULL, create_paths = NULL)
```

Arguments

`project`	Character. The path to the project directory.
`config_path`	Character. The path of the config file. Defaults to NULL.
`create_paths`	Logical indicating if missing paths should be created. Defaults to NULL.

Value

Called for its side-effects.

Examples

```
tmpdir <- tempdir()
print(tmpdir)

# account for windows
if (Sys.info()['sysname'] == "Windows") {
  tmpdir <- gsub("\\", "\\\\", tmpdir, fixed = TRUE)
}

# Create an example config file\
hierarchy <- paste0("default:
  paths:
    data: !expr list(
      DEV = file.path('",tmpdir,"', 'demo', 'DEV', 'username', 'project1', 'data'),
      PROD = file.path('",tmpdir,"', 'demo', 'PROD', 'project1', 'data'))
    output: !expr list(
      DEV = file.path('",tmpdir,"', 'demo', 'DEV', 'username', 'project1', 'output'),
      PROD = file.path('",tmpdir,"', 'demo', 'PROD', 'project1', 'output'))
    programs: !expr list(
      DEV = file.path('",tmpdir,"', 'demo', 'DEV', 'username', 'project1', 'programs'),
      PROD = file.path('",tmpdir,"', 'demo', 'PROD', 'project1', 'programs'))")


writeLines(hierarchy, file.path(tmpdir, "hierarchy.yml"))

init(project = tmpdir,
     config_path = file.path(tmpdir, "hierarchy.yml"),
     create_paths = TRUE)
```

---

library *Wrapper around library to place packages after any current autos*

Description

Autos need to immediately follow the global environment. This wrapper around `base::library()` will position any attached packages in the earliest position on the search path currently occupied by a package environment, guaranteeing newly loaded packages appear before previously loaded packages but after any currently attached non-packages.

Arguments

`...`	pass directly through to base::library
`pos`	see base::library. NULL (the default) is taken to mean the earliest position of a package environment within the current search path. If non-null, underlying behavior of base::library is respected.

Value

returns (invisibly) the list of attached packages

Examples

```
# Simple example
library(purrr)

# Illustrative example to show that autos will always remain above attached libraries
tmpdir <- tempdir()
print(tmpdir)

# account for windows
if (Sys.info()['sysname'] == "Windows") {
  tmpdir <- gsub("\\", "\\\\", tmpdir, fixed = TRUE)
}

# Create an example config file
hierarchy <- paste0("default:
  paths:
    functions: !expr list(
      DEV = file.path('",tmpdir,"', 'demo', 'DEV', 'username', 'project1', 'functions'),
      PROD = file.path('",tmpdir,"', 'demo', 'PROD', 'project1', 'functions'))
  autos:
    my_functions: !expr list(
      DEV = file.path('",tmpdir,"', 'demo', 'DEV', 'username', 'project1', 'functions'),
      PROD = file.path('",tmpdir,"', 'demo', 'PROD', 'project1', 'functions'))")


# write config
writeLines(hierarchy, file.path(tmpdir, "hierarchy.yml"))

config <- config::get(file = file.path(tmpdir, "hierarchy.yml"))

build_from_config(config)

# write function to DEV
writeLines("dev_function <- function() {print(environment(dev_function))}",
           file.path(tmpdir, 'demo/DEV/username/project1/functions/dev_function.r'))

# write function to PROD
writeLines("prod_function <- function() {print(environment(prod_function))}",
           file.path(tmpdir, 'demo/PROD/project1/functions/prod_function.r'))

# setup the environment
Sys.setenv(ENVSETUP_ENVIRON = "DEV")
rprofile(config::get(file = file.path(tmpdir, "hierarchy.yml")))

# show search
search()

# now attach purrr
library(purrr)

# see autos are still above purrr in the search path
search()
```

read_path *Read path*

Description

Check each environment for the file and return the path to the first.

Usage

```
read_path(
  lib,
  filename,
  full.path = TRUE,
  envsetup_environ = Sys.getenv("ENVSETUP_ENVIRON")
)
```

Arguments

lib	object containing the paths for all environments of a directory
filename	name of the file you would like to read
full.path	logical to return the path including the file name
envsetup_environ	name of the environment you would like to read the file from; default values comes from the value in the system variable ENVSETUP_ENVIRON which can be set by Sys.setenv(ENVSETUP_ENVIRON = "environment name")

Details

The environments searched depends on the current environment. For example, if your workflow contains a development (dev) area and production area (prod), and the code is executing in the dev environment, we search dev and prod. If in prod, we only search prod.

Value

string containing the path of the first directory the file is found

Examples

```
tmpdir <- tempdir()

# account for windows
if (Sys.info()['sysname'] == "Windows") {
  tmpdir <- gsub("\\", "\\\\", tmpdir, fixed = TRUE)
}

# add config for just the data location
hierarchy <- paste0("default:
```

```
  paths:
    data: !expr list(
      DEV = file.path('",tmpdir,"', 'demo', 'DEV', 'username', 'project1', 'data'),
      PROD = file.path('",tmpdir,"', 'demo', 'PROD', 'project1', 'data'))")

# write config file to temp directory
writeLines(hierarchy, file.path(tmpdir, "hierarchy.yml"))

config <- config::get(file = file.path(tmpdir, "hierarchy.yml"))

# build folder structure from config
build_from_config(config)

# setup environment based on config
rprofile(config::get(file = file.path(tmpdir, "hierarchy.yml")))

# place data in prod data folder
saveRDS(mtcars, file.path(tmpdir, "demo/PROD/project1/data/mtcars.rds"))

# find the location of mtcars.rds
read_path(data, "mtcars.rds")
```

`rprofile` *Function used to pass through code to the .Rprofile*

Description

Function used to pass through code to the .Rprofile

Usage

```
rprofile(config)
```

Arguments

`config`	configuration object from config::get()

Value

Called for its side effects. Directory paths and autos are added to the search path based on your config.

Examples

```
# temp location to store configuration files
tmpdir <- tempdir()
print(tmpdir)

# Create an example config file
hierarchy <- "default:
```

```
  paths:
    data: !expr list(DEV = '/demo/DEV/username/project1/data',
                     PROD = '/demo/PROD/project1/data')
    output: !expr list(DEV = '/demo/DEV/username/project1/output',
                       PROD = '/demo/PROD/project1/output')
    programs: !expr list(DEV = '/demo/DEV/username/project1/programs',
                         PROD = '/demo/PROD/project1/programs')"

writeLines(hierarchy, file.path(tmpdir, "hierarchy.yml"))

rprofile(config::get(file = file.path(tmpdir, "hierarchy.yml")))
```

`validate_config` *Validate a configuration file*

Description

A helper function to help troubleshoot common problems that can occur when building your configuration file.

Usage

```
validate_config(config)
```

Arguments

`config`	configuration object from config::get()

Value

Called for its side-effects. Prints findings from validation checks.

Examples

```
# temp location to store configuration files
tmpdir <- tempdir()
print(tmpdir)

# Each path only points to one location, i.e. there is no hierarchy for a path
no_hierarchy <- 'default:
  paths:
    data: "/demo/DEV/username/project1/data"
    output: "/demo/DEV/username/project1/output"
    programs: "/demo/DEV/username/project1/programs"'

writeLines(no_hierarchy, file.path(tmpdir, "no_hierarchy.yml"))

validate_config(config::get(file = file.path(tmpdir, "no_hierarchy.yml")))

# A path can point to multiple locations, i.e. there is a hierarchy
```

```
hierarchy <- "default:
  paths:
    data: !expr list(DEV = '/demo/DEV/username/project1/data',
                     PROD = '/demo/PROD/project1/data')
    output: !expr list(DEV = '/demo/DEV/username/project1/output',
                       PROD = '/demo/PROD/project1/output')
    programs: !expr list(DEV = '/demo/DEV/username/project1/programs',
                         PROD = '/demo/PROD/project1/programs')
    envsetup_environ: !expr Sys.setenv(ENVSETUP_ENVIRON = 'DEV'); 'DEV'"

writeLines(hierarchy, file.path(tmpdir, "hierarchy.yml"))

validate_config(config::get(file = file.path(tmpdir, "hierarchy.yml")))

# A hierarchy is present for paths, but they are not named
hierarchy_no_names <- "default:
  paths:
    data: !expr list('/demo/DEV/username/project1/data', '/demo/PROD/project1/data')
   output: !expr list('/demo/DEV/username/project1/output', '/demo/PROD/project1/output')
   programs: !expr list('/demo/DEV/username/project1/programs', '/demo/PROD/project1/programs')
    envsetup_environ: !expr Sys.setenv(ENVSETUP_ENVIRON = 'DEV'); 'DEV'"


writeLines(hierarchy_no_names, file.path(tmpdir, "hierarchy_no_names.yml"))

validate_config(config::get(file = file.path(tmpdir, "hierarchy_no_names.yml")))


# No paths are specified
no_paths <- "default:
  autos:
    my_functions: '/demo/PROD/project1/R'"

writeLines(no_paths, file.path(tmpdir, "no_paths.yml"))

validate_config(config::get(file = file.path(tmpdir, "no_paths.yml")))
```

`write_path` *Retrieve a file path from an envsetup object containing paths*

Description

Paths will be filtered to produce the lowest available level from a hierarchy of paths based on envsetup_environ

Usage

```
write_path(
  lib,
  filename = NULL,
```

```
  envsetup_environ = Sys.getenv("ENVSETUP_ENVIRON")
)
```

Arguments

lib	Object containing the paths for all environments of a directory
filename	Name of the file you would like to write
envsetup_environ	Name of the environment to which you would like to write. Defaults to the ENVSETUP_ENVIRON environment variable

Value

path to write

Examples

```
tmpdir <- tempdir()

# account for windows
if (Sys.info()['sysname'] == "Windows") {
  tmpdir <- gsub("\\", "\\\\", tmpdir, fixed = TRUE)
}

# add config for just the data location
hierarchy <- paste0("default:
  paths:
    data: !expr list(
      DEV = file.path('",tmpdir,"', 'demo', 'DEV', 'username', 'project1', 'data'),
      PROD = file.path('",tmpdir,"', 'demo', 'PROD', 'project1', 'data'))")

# write config file to temp directory
writeLines(hierarchy, file.path(tmpdir, "hierarchy.yml"))

config <- config::get(file = file.path(tmpdir, "hierarchy.yml"))

# build folder structure from config
build_from_config(config)

# setup environment based on config
rprofile(config::get(file = file.path(tmpdir, "hierarchy.yml")))

# find location to write mtcars.rds
write_path(data, "mtcars.rds")

# save data in data folder using write_path
saveRDS(mtcars, write_path(data, "mtcars.rds"))
```

Index