Package 'fail'

July 22, 2025

Type Package

Title File Abstraction Interface Layer (FAIL)

Description More comfortable interface to work with R data or source files in a key-value fashion.

Version 1.3

Author Michel Lang <michellang@gmail.com>

Maintainer Michel Lang <michellang@gmail.com>

URL https://github.com/mllg/fail

License BSD_3_clause + file LICENSE

Imports stats, utils, BBmisc, checkmate

Suggests testthat

ByteCompile yes

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-01 00:21:25

Contents

fail 2
sail 4

Index **6**

`fail` *Create a file abstraction interface layer (FAIL) object.*

Description

The general idea is to not bother about file path joining or file extensions. Instead, FAIL offers a key-value like interface to RData files in a specified directory. The filename (without extension) acts as the key while the stored R objects are the values. Fail provides an interface to the basic file system actions: listing, reading / loading, writing / saving, removing and applying functions on files. An implemented cache mechanism can be used to avoid repeated disk reads.

Usage

```
fail(path = getwd(), extension = "RData", all.files = FALSE,
  use.cache = FALSE, simplify = TRUE)
```

Arguments

`path`	[`character(1)`] Path to work in, will be created if it does not exists.
`extension`	[`character(1)`] File extension to work with. Default is "RData".
`all.files`	[`logical(1)`] Also include hidden files, i.e. files whose name start with a dot (“.”). Default is `FALSE`.
`use.cache`	[`logical(1)`] Use a memory cache per global default. Global option which can locally be overwritten in most functions. Default is `FALSE`
`simplify`	[`character(1)`] If only one object is stored in a R data file, should the return value be simplified? If set to `TRUE`, instead of a list containing one element the object itself will be returned.

Details

For a quick introduction on the usage, see https://github.com/mllg/fail.

An object with the following functions is returned by `fail`:

`ls(pattern=NULL)` Function to list keys in directory `path` matching a regular expression pattern `pattern`. Returns a character vector of keys.

`get(key, use.cache)` Function to load a file identified by `key` from directory `path`. To load many objects at once, use `as.list`, `assign` or `get` together with `lapply`. Argument `use.cache` can be set to temporarily overwrite the global `use.cache` flag.

`put(..., li, keys, use.cache)` Function to save objects to directory `path`. Names for objects provided via `...` will be looked up or can be provided using a `key = value` syntax. More objects can be passed as a named list using the argument `li`: Each list item will be saved to a separate file. If you provide `keys` as a character vector, these names will be taken for the arguments passed via `....` Argument `use.cache` temporarily overwrites the global `use.cache` flag. Returns a character vector of stored keys.

`remove(keys)` Function to remove files identified by `keys` from directory `path`. Returns a character vector of deleted keys.

`apply(FUN, ..., keys, use.cache, simplify=FALSE, use.names=TRUE)` Apply function `FUN` on files identified by `keys`. `keys` defaults to all keys available and will be used to name the returned list. The loaded R objects will be past unnamed as first argument. Use `...` for additional function arguments. Argument `use.cache` can be set to temporarily overwrite the global `use.cache` flag. For arguments `simplify` and `use.names`, see `lapply`.

`mapply(FUN, ..., keys, use.cache, moreArgs = NULL, simplify=FALSE, use.names=TRUE)` Apply function `FUN` on files identified by `keys`. `keys` defaults to all keys available and will be used to name the returned list. The function `FUN` must have the formal arguments "key" and "value". Both key and value will be passed named. Use `...` and/or `moreArgs` for additional function arguments. Argument `use.cache` can be set to temporarily overwrite the global `use.cache` flag. For arguments `moreArgs`, `simplify` and `use.names`, see `mapply`.

`as.list(keys, use.cache)` Return a named list of objects identified by `keys`. `keys` defaults to all keys available. Argument `use.cache` can be set to temporarily overwrite the global `use.cache` flag.

`assign(keys, envir=parent.frame(), use.cache)` Assigns all objects identified by the character vector `keys` in the environment `envir`. Argument `use.cache` can be set to temporarily overwrite the global `use.cache` flag. Returns a character vector of assigned keys.

`clear(keys)` Clear the cache to free memory. `keys` defaults to all keys available. Returns a character vector of cleared keys.

`cached()` Returns a character vector of keys of cached objects.

`size(keys, unit="b")` Get the file size in Bytes of the files identified by `keys`. `keys` defaults to all keys available. Argument `unit` accepts "b", "Kb", "Mb" and "Gb" and can be used to convert Bytes to KiloBytes, MegaBytes or GigaBytes, respectively.

`info()` Returns a named list with `path`, `extension` and `use.cache`. Internally used for the `print` method with a much nicer summary of the FAIL object.

Furthermore, the package provides S3 methods for `print` and `as.list`.

Be aware of the following restriction regarding file names and keys: The package performs some basic checks for illegal characters on the key names. In principle all characters matching the pattern "[a-zA-Z0-9._-]" are allowed and should work on most or all file systems. But be careful with key names which are not compatible with R's variable naming restrictions, e.g. using the minus character or key names starting with a number: these provoke unwanted side effects and will result in errors if used with `assign`.

If two files would collide on case-insensitive file systems like Windows' NTFS, the package will throw warnings. Best practice is to not rely on case sensitivity.

Value

Object of class `fail`. See details.

Examples

```
# initialize a FAIL in a temporary directory
path <- tempfile("")
files <- fail(path)

# save x and y, vectors of random numbers
x <- runif(100)
files$put(x, y = runif(100))

# save columns of the iris data set as separate files
files$put(li = as.list(iris))

# load all RData files in a named list as a one-liner
as.list(fail(path))

# load a single object from the file system
files$get("Species")
files$as.list(c("x", "y"))

# remove an object (and related file)
files$remove("Species")

# apply a function over files
files$apply(mean)
files$mapply(function(key, value) sprintf("%s -> %f", key, mean(value)), simplify = TRUE)

# show file size informations
files$size(unit = "Mb")

# get an object and cache it
files$get("x", use.cache = TRUE)
files$cached()
files$clear()
files$cached()

# assign variables in the current environment
files$assign("y")
mean(y)
```

`sail` *Create a source abstraction interface layer (SAIL) object.*

Description

This function returns an object of class `sail` which behaves like `fail`, but is indented for loading and saving R source code files.

Usage

```
sail(path = getwd(), extension = "R", all.files = FALSE,
  use.cache = FALSE, simplify = TRUE, suppressMessages = FALSE)
```

Arguments

path	[character(1)] Path to work in, will be created if it does not exists.
extension	[character(1)] File extension to work with. Default is "R".
all.files	[logical(1)] Also include hidden files, i.e. files whose name start with a dot (".."). Default is FALSE.
use.cache	[logical(1)] Use a memory cache per global default. Global option which can locally be overwritten in most functions. Default is FALSE
simplify	[character(1)] If only one object is defined in a sourced R file, should the return value be simplified? If set to TRUE, instead of a list containing one element the object itself will be returned.
suppressMessages	[logical(1)] Wrap the sys.source command into suppressMessages and link[base]{suppressPackageStartupM Default is FALSE, i.e. you will see regular output of sourced scripts.

Value

Object of class sail. See the documentation of fail for details.

Index

as.list, *3*

fail, 2, *4, 5*

lapply, *2, 3*

mapply, *3*

print, *3*

sail, 4
suppressMessages, *5*
sys.source, *5*