Package: kwb.file (via r-universe)

September 9, 2024

Title Functions Related to File and Path Operations

Version 0.3.2

Description This package provides helper functions that have been developed during different research projects at KWB. The functions are dealing with file operations and handling file and folder paths. Let's see what we have in different scripts and other packages and better fits here...

License MIT + file LICENSE

URL https://github.com/KWB-R/kwb.file

BugReports https://github.com/KWB-R/kwb.file/issues

Imports digest, dplyr, fs, kwb.utils (>= 0.6.0), yaml

Suggests covr, testthat

Remotes github::kwb-r/kwb.utils

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

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

RemoteUrl https://github.com/KWB-R/kwb.file

RemoteRef HEAD

RemoteSha 87a026edc457445dd3a0e5f03d6072bd845e86c3

Contents

`add_file_info` *Add File Information From File Database*

Description

Add File Information From File Database

Usage

```
add_file_info(data)
```

Arguments

`data`	data frame with column `file_id` containing file identifiers and with an attribute `file_db` containing a "file database" as created by `to_file_database`

Value

data frame `data` with additional columns `folder_path` and `file_name`

Examples

```
# Define some paths
paths <- c(
  "/very/long/path/very_long_file_name_1",
  "/very/long/path/very_long_file_name_2",
  "/very/long/path/very_long_file_name_3"
)

# Create a "file database" from the paths
file_db <- kwb.file::to_file_database(paths, remove_common_base = FALSE)

# Create a data frame that relates some information to the files.
# Use the file identifier instead of the full name to keep the data clean
(df <- kwb.utils::noFactorDataFrame(
  file_id = file_db$files$file_id,
  value = seq_along(paths)
))

# Store the file database in the attribute "file_db"
df <- structure(df, file_db = file_db)
```

```
# Restore the full file paths
add_file_info(df)
```

---

copy_files_to_target_dir *Copy Files to Flat Structure*

Description

Calls `file.copy` under the hood but gives a message about the indices and paths of the files that could not be copied.

Usage

```
copy_files_to_target_dir(from_paths, target_dir, target_files)
```

Arguments

`from_paths`	paths to the files to be copied
`target_dir`	path to the target directory
`target_files`	relative paths to the target files, relative to `target_dir`

Examples

```
root <- system.file(package = "kwb.file")

relative_paths <- dir(root, recursive = TRUE)

# The original files are in root or in different subfolders
relative_paths

# Create a temporary target folder
target_dir <- kwb.utils::createDirectory(file.path(tempdir(), "target"))

# Copy all files into one target folder without subfolders
from_paths <- file.path(root, relative_paths)
to_paths <- basename(from_paths)

# Make sure that the target file names contain no duplicates, otherwise
# an error is raised
to_paths <- kwb.utils::makeUnique(to_paths, warn = FALSE)

# Copy the files
copy_files_to_target_dir(from_paths, target_dir, to_paths)

# Look at the result
dir(target_dir, recursive = TRUE)
```

dir_full *Helper function to return full paths*

Description

This function provides a shortcut to `dir(..., full.names = TRUE)`

Usage

```
dir_full(...)
```

Arguments

`...`	arguments passed to `dir`

Examples

```
dir_full(system.file(package = "kwb.file"))
```

dir_full_recursive_xml *Get Full Paths to all XML files Below a Root Folder*

Description

Get Full Paths to all XML files Below a Root Folder

Usage

```
dir_full_recursive_xml(root)
```

Arguments

`root`	path to root folder

Value

vector of character

get_download_dir *Get Default Download Directory*

Description

Get Default Download Directory

Usage

```
get_download_dir()
```

Value

assumed default download directory on the user's computer (vector of character of length one)

Examples

```
dir_full(get_download_dir())
```

read_file_metadata *Read File Metadata from YAML-File*

Description

Read File Metadata from YAML-File

Usage

```
read_file_metadata(
  yaml_file,
  file_encoding = "UTF-8",
  out_class = c("data.frame", "list")[1]
)
```

Arguments

yaml_file	path to YAML-File containing file metadata (as saved with kwb.file:::write_file_info_to_yaml_fi
file_encoding	passed to argument fileEncoding of read_yaml
out_class	one of "data.frame", "list"

Value

depending on `out_class`, either a data frame with the following columns or a list with the following elements is returned:

file_id clean file name given to the original file for simpler access,

original_name original file name given by data provider,

original_folder original path to folder in which file was provided.

`remove_common_root` *Remove the Common Root Parts*

Description

Remove the Common Root Parts

Usage

```
remove_common_root(x, n_keep = 1L, dbg = TRUE)
```

Arguments

`x`	list of vectors of character as returned by `strsplit` or a vector of character.
`n_keep`	minimum number of segments to be kept in any case in the returned relative paths. For example, two paths "a" and "a/b" have the common root "a". Removing this root would result in relative paths "" and "b". As this is not useful, `n_keep` is 1 by default, making sure that all paths keep at least one segment (segment "a") in the example.
`dbg`	if `TRUE` debug messages are shown

Examples

```
# Split paths at the slashes
absparts <- strsplit(c("a/b/c", "a/b/d", "a/b/e/f/g", "a/b/hi"), "/")

# Remove the common parts of the paths
relparts <- remove_common_root(absparts)
relparts

# The extracted root is returned in attribute "root"
attr(relparts, "root")
```

split_into_dir_and_file *"Split Full Paths into Directory Path and Filename"*

Description

"Split Full Paths into Directory Path and Filename"

Usage

```
split_into_dir_and_file(paths)
```

Arguments

paths	vector of character representing full file paths

Value

data frame with columns `directory` and `file`

Examples

```
split_into_dir_and_file(c("path/to/file-1", "path/to/file-2"))
```

split_into_root_folder_file_extension *Split Full Paths into Root, Folder, File and Extension*

Description

Split Full Paths into Root, Folder, File and Extension

Usage

```
split_into_root_folder_file_extension(paths, n_root_parts = 0)
```

Arguments

paths	vector of character representing full file paths
n_root_parts	number of first path segments considered as "root"

Value

data frame with columns `root`, `folder`, `file`, `extension`, `depth`

Examples

```
paths <- c(
  "//always/the/same/root/project-1/intro.doc",
  "//always/the/same/root/project-1/logo.png",
  "//always/the/same/root/project-2/intro.txt",
  "//always/the/same/root/project-2/planning/file-1.doc",
  "//always/the/same/root/project-2/result/report.pdf"
)

split_into_root_folder_file_extension(paths)
split_into_root_folder_file_extension(paths, n_root_parts = 6)
split_into_root_folder_file_extension(paths, n_root_parts = 7)
```

---

split_paths *Split Full Paths at Slashes into Parts*

Description

Split Full Paths at Slashes into Parts

Usage

```
split_paths(paths, dbg = TRUE, use_fs = FALSE)
```

Arguments

paths	vector of character representing full file paths
dbg	if TRUE (default), a debug message is shown
use_fs	whether or not to simply use path_split. Defaults to FALSE

Examples

```
segments <- split_paths(c("path/to/file-1", "path/to/file-2"))
segments
```

---

to_file_database *Create Two Table Relational Database From Paths*

Description

From a vector of given file paths, this function generates short and unique identifiers for files and folders. The assignements between identifiers and original paths are stored in two data frames, files and folders that are returned.

Usage

```
to_file_database(files, remove_common_base = TRUE)
```

Arguments

files	vector of file paths
remove_common_base	if TRUE (default) the common root of all files is removed before creating the database

Value

list of two data frames, files and folders

Examples

```
paths <- c(
  "very_long/very_ugly_path/even with spaces.doc",
  "very_long/very_ugly_path/even with spaces.docx"
)

to_file_database(paths)
to_file_database(paths, remove_common_base = FALSE)
```

---

to_simple_names *Convert Long File Paths to Simple Paths*

Description

Convert Long File Paths to Simple Paths

Usage

```
to_simple_names(paths, method = 1L, get_base = NULL, sha1_digits = 4)
```

Arguments

paths	vector of character containing file paths
method	method = 1: file names generated match the pattern file_<xx> with <xx> being an integer number of two digits. method = 2: file names generated match the pattern file_<sha> with <sha> being the first sha1_digits digits of the sha1 hash (see e.g. http://www.sha1-online.com/) of the base names of the paths. By default, the base name is the file name (without folder path) without extension. The base names can be determined individually by providing a function in get_base

get_base	function taking a vector of character as input and returning a vector of character as output. If not NULL, this function will be used to determine the base paths from the paths when method = 2 was specified.
sha1_digits	number of digits used when method = 2 is to be applied

Value

vector of character as long as paths

Examples

```
paths <- c("v1_ugly_name_1.doc",  "v1_very_ugly_name.xml",
           "v2_ugly_name_1.docx", "v2_very_ugly_name.xmlx")

to_simple_names(paths, method = 1L)
writeLines(sort(to_simple_names(paths, method = 2L)))

# All sha1 are different because all base names (file name without extension
# by default) are different. If you want to give the same sha1 to files that
# correspond to each other but have a different extension, set the function
# that extracts the "base name" of the file:

get_base <- function(x) kwb.utils::removeExtension(gsub("^v\\d+_", "", x))

writeLines(sort(to_simple_names(paths, method = 2L, get_base = get_base)))

# Now the file names that have the same base name (neglecting the prefix
# v1_ or v2_) get the same sha1 and thus appear as groups in the sorted
# file list
```

---

to_subdir_matrix *Convert a Vector of Paths to a Matrix of Subfolders*

Description

Convert a Vector of Paths to a Matrix of Subfolders

Usage

```
to_subdir_matrix(
  paths,
  fill.value = "",
  result_type = "matrix",
  dbg = FALSE,
  method = NA_integer_
)
```

Arguments

paths	vector of path strings
fill.value	value used to fill empty cells of the result matrix
result_type	one of c("matrix", "data.frame", "list"), specifying the type of object to be returned. Result type "list" is only implemented for method = 2.
dbg	if TRUE debug messages are shown
method	integer specifying the implementation method. Currently not used.

Value

matrix or data frame, depending on result_type

Examples

```
folder_matrix <- kwb.file::to_subdir_matrix(c("a1/b1/c1", "a1/b2", "a2"))

folder_matrix

dim(folder_matrix)

folder_matrix[folder_matrix[, 1] == "a1", ]
```

Index