Package: findInFiles (via r-universe)

August 26, 2024

Type Package

Title Find Pattern in Files

Version 0.5.0

Description Creates a HTML widget which displays the results of searching for a pattern in files in a given folder. The results can be viewed in the 'RStudio' viewer pane, included in a 'R Markdown' document or in a 'Shiny' application. Also provides a 'Shiny' application allowing to run this widget and to navigate in the files found by the search. Instead of creating a HTML widget, it is also possible to get the results of the search in a 'tibble'. The search is performed by the 'grep' command-line utility.

License GPL-3

URL `https://github.com/stla/findInFiles`

BugReports `https://github.com/stla/findInFiles/issues`

Imports crayon, htmlwidgets, shiny, stringi, stringr, tibble, utils, vctrs

Suggests fs, shinyAce, shinyFiles, shinyjqui, shinyvalidate, shinyWidgets

Encoding UTF-8

RoxygenNote 7.3.1

SystemRequirements grep

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

RemoteUrl https://github.com/stla/findinfiles

RemoteRef HEAD

RemoteSha 2ef6d8c09051ee905ce6a612aa94b78ce149dae9

Contents

FIF2dataframe *Output of 'findInFiles' as a dataframe*

Description

Returns the results of `findInFiles` in a dataframe, when the option `output = "viewer+tibble"` or `output = "tibble"` is used.

Usage

```
FIF2dataframe(fif)
```

Arguments

`fif`	the output of `findInFiles` used with the option `output = "viewer+tibble"` or `output = "tibble"`

Value

The results of `findInFiles` in a dataframe.

Examples

```
folder <- system.file("example", package = "findInFiles")
fif <- findInFiles("R", "function", root = folder, output = "viewer+tibble")
FIF2dataframe(fif)
fif
```

FIF2tibble *Output of ‘findInFiles‘ as a tibble*

Description

Returns the results of findInFiles in a tibble, when the option `output = "viewer+tibble"` is used.

Usage

```
FIF2tibble(fif)
```

Arguments

`fif`	the output of findInFiles used with the option `output = "viewer+tibble"`

Value

The results of findInFiles in a tibble.

Examples

```
folder <- system.file("example", package = "findInFiles")
fif <- findInFiles("R", "function", root = folder, output = "viewer+tibble")
FIF2tibble(fif)
fif
```

findInFiles *Find pattern in files*

Description

Find a pattern in some files. The functions `findInFiles` and `fif` are the same, and `fifR(...)` is the same as `findInFiles(extensions = "R", ...)`.

Usage

```
findInFiles(
  extensions,
  pattern,
  depth = NULL,
  maxCountPerFile = NULL,
  maxCount = NULL,
  wholeWord = FALSE,
  ignoreCase = FALSE,
  extended = FALSE,
  fixed = FALSE,
  perl = FALSE,
  includePattern = NULL,
  excludePattern = NULL,
  excludeFoldersPattern = NULL,
  moreOptions = NULL,
  root = ".",
  output = "viewer",
  elementId = NULL
)

fif(
  extensions,
  pattern,
  depth = NULL,
  maxCountPerFile = NULL,
  maxCount = NULL,
  wholeWord = FALSE,
  ignoreCase = FALSE,
  extended = FALSE,
  fixed = FALSE,
  perl = FALSE,
  includePattern = NULL,
  excludePattern = NULL,
  excludeFoldersPattern = NULL,
  moreOptions = NULL,
  root = ".",
  output = "viewer",
  elementId = NULL
)

fifR(...)
```

Arguments

`extensions`	extension(s) of the files to include in the search (case-sensitive), e.g. `"R"` or `c("R", "Rmd")`, or `"*"` to search in all files
`pattern`	pattern to search for, a regular expression, e.g. `"function"` or `"^function"`, or a string if `fixed=TRUE`; by default the pattern is considered as a basic regular expression, but this can be changed to an extended regular expression by setting `extended=TRUE` or to a Perl regular expression by setting `perl=TRUE`
`depth`	depth of the search, `NULL` or a negative number for an entire recursive search (subdirectories, subdirectories of subdirectories, etc.), otherwise a positive integer: `0` to search in the root directory only, `1` to search in the root directory and its subdirectories, etc.
`maxCountPerFile`	maximum number of results per file, `NULL` for an unlimited number, otherwise a positive integer; when an integer `m` is supplied, `grep` stops to search in each file

	after it finds `m` results
`maxCount`	maximum number of results, `NULL` for an unlimited number, otherwise a positive integer; supplying an integer `m` just truncates the output, it does not stop `grep` after `m` results are found (so there is no gain of efficiency)
`wholeWord`	logical, whether to match the whole pattern
`ignoreCase`	logical, whether to ignore the case
`extended`	logical, whether the pattern given in the `pattern` is an extended regular expression; if `TRUE`, you can search for multiple patterns by passing a string like `"(pattern1|pattern2|...)"` to the `pattern` argument
`fixed`	logical, whether the pattern given in the `pattern` argument is a string to be matched as is, or, to search for multiple patterns, multiple strings separated by `"\n"`
`perl`	logical, whether the pattern given in the `pattern` argument is a Perl regular expression; if `TRUE`, you can search for multiple patterns by passing a string like `"(pattern1|pattern2|...)"` to the `pattern` argument
`includePattern`	this argument is ignored if `depth` is not a positive integer; it must be a pattern or a vector of patterns, and only the files whose name matches this pattern or one of these patterns will be included in the search
`excludePattern`	a pattern or a vector of patterns; files and folders whose name matches this pattern or one of these patterns will be excluded from search
`excludeFoldersPattern`	a pattern or a vector of patterns; folders whose name matches this pattern or one of these patterns will be excluded from search
`moreOptions`	additional options passed to the `grep` command, for `grep` experts
`root`	path to the root directory to search from
`output`	one of `"viewer"`, `"tibble"` or `"viewer+tibble"`; set `"tibble"` to get a tibble, `"viewer"` to get a `htmlwidget`, and `"viewer+tibble"` to get a `htmlwidget` from which you can extract a tibble with the function `FIF2tibble`
`elementId`	a HTML id, usually useless
`...`	arguments other than `extensions` passed to `findInFiles`

Value

A tibble if `output="tibble"`, otherwise a `htmlwidget` object.

Examples

```
library(findInFiles)
folder <- system.file("example", package = "findInFiles")
findInFiles("R", "function", root = folder)

findInFiles("R", "function", root = folder, output = "tibble")

fif <- findInFiles("R", "function", root = folder, output = "viewer+tibble")
FIF2tibble(fif)
```

```
FIF2dataframe(fif)
fif

folder <- system.file("www", "shared", package = "shiny")
findInFiles(
  "css", "color", root = folder,
  excludePattern = c("*.min.css", "selectize*", "shiny*")
)
```

`findInFiles-shiny` *Shiny bindings for 'findInFiles'*

Description

Output and render functions for using `findInFiles` within Shiny applications and interactive Rmd documents.

Usage

```
FIFOutput(outputId, width = "100%", height = "400px")

renderFIF(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId`	output variable to read from
`width, height`	a valid CSS unit (like "100%", "400px", "auto") or a number, which will be coerced to a string and have "px" appended
`expr`	an expression that generates a 'findInFiles' widget
`env`	the environment in which to evaluate `expr`
`quoted`	logical, whether `expr` is a quoted expression (with `quote()`)

Value

`FIFOutput` returns an output element that can be included in a Shiny UI definition, and `renderFIF` returns a `shiny.render.function` object that can be included in a Shiny server definition.

Examples

```
library(findInFiles)
library(shiny)

onKeyDown <- HTML(
  'function onKeyDown(event) {',
  '  var key = event.which || event.keyCode;',
  '  if(key === 13) {',
  '    Shiny.setInputValue(',
  '      "pattern", event.target.value, {priority: "event"}',
  '    );',
  '  }',
  '}'
)

ui <- fluidPage(
  tags$head(tags$script(onKeyDown)),
  br(),
  sidebarLayout(
    sidebarPanel(
      selectInput(
        "ext", "Extension",
        choices = c("R", "js", "css")
      ),
      tags$div(
        class = "form-group shiny-input-container",
        tags$label(
          class = "control-label",
          "Pattern"
        ),
        tags$input(
          type = "text",
          class = "form-control",
          onkeydown = "onKeyDown(event);",
          placeholder = "Press Enter when ready"
        )
      ),
      numericInput(
        "depth", "Depth (set -1 for unlimited depth)",
        value = 0, min = -1, step = 1
      ),
      checkboxInput(
        "wholeWord", "Whole word"
      ),
      checkboxInput(
        "ignoreCase", "Ignore case"
      )
    ),
    mainPanel(
      FIFOutput("results")
    )
  )
)


server <- function(input, output){

  output[["results"]] <- renderFIF({
    req(input[["pattern"]])
    findInFiles(
      extensions = isolate(input[["ext"]]),
      pattern    = input[["pattern"]],
      depth      = isolate(input[["depth"]]),
      wholeWord  = isolate(input[["wholeWord"]]),
      ignoreCase = isolate(input[["ignoreCase"]])
    )
  })

}

if(interactive()){
  shinyApp(ui, server)
}
```

`shinyFIF` *Shiny application 'Find in files'*

Description

Launches a Shiny application allowing to run `findInFiles` and to navigate in the results.

Usage

```
shinyFIF()
```

Value

No returned value, just launches the Shiny application.

Note

The packages listed in the **Suggests** field of the package description are required.

Index