

Package: findInGit (via r-universe)

August 20, 2024

Type Package

Title Find Pattern in Files of All Branches of a 'git' Repository

Version 0.1.1

Description Creates a HTML widget which displays the results of searching for a pattern in files in a given 'git' repository, including all its branches. The results can also be returned in a dataframe.

License GPL-3

Encoding UTF-8

SystemRequirements grep, git

Imports stringr, crayon, htmlwidgets

Suggests shiny, R.utils

URL <https://github.com/stla/findInGit>

BugReports <https://github.com/stla/findInGit/issues>

RoxygenNote 7.1.1

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

RemoteUrl <https://github.com/stla/findingit>

RemoteRef HEAD

RemoteSha 0056f8f43909d9f683e3348f968ef404b345fb24

Contents

FIG2dataframe	2
findInGit	2
findInGit-shiny	4

Index	8
--------------	----------

FIG2dataframe	<i>Output of 'findInGit' as a dataframe</i>
---------------	---

Description

Returns the results of `findInGit` in a dataframe, when the option `output = "viewer+dataframe"` is used. See the example in `findInGit`.

Usage

```
FIG2dataframe(fig)
```

Arguments

`fig` the output of `findInGit` used with the option `output = "viewer+dataframe"`

Value

The results of `findInGit` in a dataframe.

findInGit	<i>Find pattern in files of a 'git' repository</i>
-----------	--

Description

Find a pattern in the files with a given extension, in all branches of a 'git' repository.

Usage

```
findInGit(  
  ext,  
  pattern,  
  wholeWord = FALSE,  
  ignoreCase = FALSE,  
  perl = FALSE,  
  excludePattern = NULL,  
  excludeFoldersPattern = NULL,  
  root = ".",  
  output = "viewer"  
)
```

Arguments

ext	file extension, e.g. "R" or "js"
pattern	pattern to search for, a regular expression, e.g. "function" or "^function"
wholeWord	logical, whether to match the whole pattern
ignoreCase	logical, whether to ignore the case
perl	logical, whether pattern is a Perl regular expression
excludePattern	a pattern; exclude from search the files and folders which match this pattern
excludeFoldersPattern	a pattern; exclude from search the folders which match this pattern
root	path to the root directory to search from
output	one of "viewer", "dataframe" or "viewer+dataframe"; see examples

Value

A dataframe if output="dataframe", otherwise a htmlwidget object.

Examples

```
findGit <- Sys.which("git") != ""
if(findGit){

  library(findInGit)
  library(R.utils) # to use the `copyDirectory` function
  folder1 <- system.file("htmlwidgets", package = "findInGit")
  folder2 <- system.file("htmlwidgets", "lib", package = "findInGit")
  tmpDir <- paste0(tempdir(), "_gitrepo")
  dir.create(tmpDir)
  # set tmpDir as the working directory
  cd <- setwd(tmpDir)
  # copy folder1 in tmpDir
  copyDirectory(folder1, recursive = FALSE)
  # initialize git repo
  system("git init")
  # add all files to git
  system("git add -A")
  # commit files
  system('git commit -m "mycommit1"')
  # create a new branch
  system("git checkout -b newbranch")
  # copy folder2 in tmpDir, under the new branch
  copyDirectory(folder2, recursive = FALSE)
  # add all files to git
  system("git add -A")
  # commit files
  system('git commit -m "mycommit2"')

  # now we can try `findInGit`
  findInGit(ext = "js", pattern = "ansi")
}
```

```
# get results in a dataframe:
findInGit(ext = "js", pattern = "ansi", output = "dataframe")

# one can also get the widget and the dataframe:
fig <- findInGit(ext = "css", pattern = "color", output = "viewer+dataframe")
fig
FIG2dataframe(fig)

# return to initial current directory
setwd(cd)
# delete tmpDir
unlink(tmpDir, recursive = TRUE, force = TRUE)

}
```

findInGit-shiny

Shiny bindings for findInGit

Description

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

Usage

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

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

Arguments

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

Value

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

Examples

```

findGit <- Sys.which("git") != ""
if(findGit){

  library(findInGit)
  library(shiny)

  # First, we create a temporary git repo
  library(R.utils) # to use the `copyDirectory` function
  folder1 <- system.file("htmlwidgets", package = "findInGit")
  folder2 <- system.file("htmlwidgets", "lib", package = "findInGit")
  tmpDir <- paste0(tempdir(), "_gitrepo")
  dir.create(tmpDir)
  # set tmpDir as the working directory
  cd <- setwd(tmpDir)
  # copy folder1 in tmpDir
  copyDirectory(folder1, recursive = FALSE)
  # initialize git repo
  system("git init")
  # add all files to git
  system("git add -A")
  # commit files
  system('git commit -m "mycommit1"')
  # create a new branch
  system("git checkout -b newbranch")
  # copy folder2 in tmpDir, under the new branch
  copyDirectory(folder2, recursive = FALSE)
  # add all files to git
  system("git add -A")
  # commit files
  system('git commit -m "mycommit2"')

  # Now let's play with 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("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"
      )
    ),
    checkboxInput(
      "wholeWord", "Whole word"
    ),
    checkboxInput(
      "ignoreCase", "Ignore case"
    )
  ),
  mainPanel(
    FIGOutput("results")
  )
)
)

Clean <- function(){
  setwd(cd)
  unlink(tmpDir, recursive = TRUE, force = TRUE)
}

server <- function(input, output){

  onSessionEnded(Clean)

  output[["results"]] <- renderFIG({
    req(input[["pattern"]])
    findInGit(
      ext = isolate(input[["ext"]]),
      pattern = input[["pattern"]],
      wholeWord = isolate(input[["wholeWord"]]),
      ignoreCase = isolate(input[["ignoreCase"]])
    )
  })
}

if(interactive()){
  shinyApp(ui, server)
}else{
  Clean()
}

```

}

Index

FIG2dataframe, [2](#)
FIGOutput (findInGit-shiny), [4](#)
findInGit, [2](#), [2](#), [4](#)
findInGit-shiny, [4](#)
renderFIG (findInGit-shiny), [4](#)