

Package ‘autoshiny’

May 7, 2026

Title Automatic Transformation of an 'R' Function into a 'shiny' App

Version 0.0.3

Description Static code compilation of a 'shiny' app given an R function (into 'ui.R' and 'server.R' files or into a 'shiny' app object). See examples at <<https://github.com/alekrutkowski/autoshiny>>.

URL <https://github.com/alekrutkowski/autoshiny>

Depends R (>= 3.4.0)

License GPL-2

Encoding UTF-8

RoxygenNote 7.2.3

Imports shiny, utils

Suggests roxygen2, magrittr, webshot

NeedsCompilation no

Author Aleksander Rutkowski [aut, cre]

Maintainer Aleksander Rutkowski <alek.rutkowski@gmail.com>

Repository CRAN

Date/Publication 2023-03-09 10:50:05 UTC

Contents

File	2
make	2
Index	6

File	<i>An obligatory wrapper for file names (paths)</i>
------	---

Description

This function **must** be used

- in the **arguments** of function fun (passed to `makeApp` or `makeFiles`) and/or
- in the **value returned** by fun

to wrap the character string indicating a path respectively

- to an input file ("consumed" by fun or
- to an output file ("produced" by fun as a **side effect**).

Otherwise **autoshiny** cannot distinguish file paths from character strings.

Usage

```
File(x)
```

Arguments

x	A string, i.e. character vector of length 1, indicating a file path to an existing file.
---	--

Value

x with an S3 class attribute "file".

make	<i>Create a Shiny app (object or files) from an R function</i>
------	--

Description

Create a Shiny app (object or files) from an R function

Usage

```
makeApp(fun, withGoButton = FALSE)
```

```
makeFiles(fun, withGoButton = FALSE, directory)
```

Arguments

fun	A function (preferably a symbol – a long self-explanatory name – pointing to a pre-defined function object, rather than an anonymous function) with zero or more arguments/parameters. Every argument must have a default value , which will be used to define each argument's: <ul style="list-style-type: none"> • type/class, • allowed values, • pre-selected/start-up value.
withGoButton	Either TRUE or FALSE (default: FALSE). It indicates if the (re)evaluation of fun in the Shiny app should be immediately triggered by every change in the value of any argument/parameter (withGoButton = FALSE) or if the (re)calculation should be started only when a specific button is pressed (withGoButton = TRUE). The latter is preferred if the (re)evaluation of fun is significantly time-consuming or if fun has no arguments (because then, without the button, only refreshing the web page would trigger the (re)evaluation).
directory	Path to a directory/folder where makeFiles should save the compiled server .R and ui .R files.

Value

makeApp A Shiny app object as returned by [as.shiny.appobj](#).

makeFiles NULL. This function saves two plain text files: ui.R and server.R with the R code of function fun translated into a Shiny app. If these files need further manual changes, it is recommended that they are first re-formatted e.g. in RStudio (top menu -> Code -> Reformat Code or Ctrl+Shift+A) or programmatically (e.g. <https://github.com/google/rfmt>).

Examples

```
## Not run:
library(shiny)

### Example 1: Trivial anonymous function
makeApp(function(x=1:3, y=5:9) x+y)

### Example 2: Nicer function and argument names
`Histogram for normal distribution` <-
  function(`Number of observations` =
    # as.integer => the argument interpreted as categorical:
    as.integer(c(100,10,1000)))
    # Generic R plots as "return values" are supported:
    plot(hist(rnorm(`Number of observations`)))
makeApp(`Histogram for normal distribution`)

### Example 3: Data frame in (upload CSV), data frame out (displayed and downloadable as CSV)
`Table of sin and cos values` <-
  function(`Upload CSV file with column "x"` =
    data.frame(x = seq(0, 2*pi, .25))) {
    dta <- `Upload CSV file with column "x"`
    data.frame(X = dta$x,
```

```

        `Sin of X` = sin(dta$x),
        `Cos of X` = cos(dta$x),
        check.names = FALSE)
    }
makeApp(`Table of sin and cos values`)

### Example 4: Arbitrary input and output files
openxlsx::write.xlsx(data.frame(x=1:5,
                                y=11:15),
                    'my_test_file.xlsx')
`Excel file in and out` <-
  function(`Input Excel file` =
    File('my_test_file.xlsx')) { # File() obligatory here!
    my.data <- openxlsx::read.xlsx(`Input Excel file`)
    my.data2 <- within(my.data,
                      z <- x + y)
    openxlsx::write.xlsx(my.data2,
                        'my_test_file_2.xlsx')
    File('my_test_file_2.xlsx') # File() obligatory here too!
  }
makeApp(`Excel file in and out`)

### Example 5: Using a button as a (re-)evaluation trigger
### Use this option if:
### - the evaluation of your function takes time, so it should not be re-evaluated with every
###   minor change of the value of inputs/arguments/parameter;
### - the function is impure e.g. depends on some external data fetched internally and takes no
###   arguments/parameters -- in such a case the function would be re-evaluated only through
###   page refresh of the browser; the button is a faster and a more elegant solution.
`Get "GDP and main components" from Eurostat` <-
  function() {
    # Getting data from
    # http://ec.europa.eu/eurostat/estat-navtree-portlet-prod/BulkDownloadListing ...
    # ... ?sort=1&file=data%2Fnama_10_gdp.tsv.gz
    x <- eurodata::importData('nama_10_gdp')
    head(x, 10)
  }
makeApp(`Get "GDP and main components" from Eurostat`,
      withGoButton = TRUE)

### Example 6: Lists of inputs (arguments) and the output list (composite return value)
### are always decomposed
`A function with lists everywhere` <-
  function(`First argument group`,` = list(`number one` = 1:3,
                                           `number two` = letters[1:3]),
          `2nd arg group`,` = list(`1st argument` = 11:14,
                                   `second arg.` = LETTERS[1:5]))
  list(`Some text` =
    as.character(c(`First argument group`,``number two`,
                  `2nd arg group`,``second arg.`)),
    `Some numbers` =
    `First argument group`,``number one` +
    `2nd arg group`,``1st argument`,

```

```
      `Even a ggplot2 chart` =  
        ggplot2::qplot(a,b,data=data.frame(a=1:20,b=log(1:20)))  
makeApp(`A function with lists everywhere`)  
  
## End(Not run)
```

Index

`as.shiny.appobj`, 3

`File`, 2

`make`, 2

`makeApp`, 2

`makeApp (make)`, 2

`makeFiles`, 2

`makeFiles (make)`, 2