Package: editData (via r-universe)

September 2, 2024

Type Package

Title 'RStudio' Addin for Editing a 'data.frame'

Version 0.2.1

Imports shiny (>= 0.13), miniUI (>= 0.1.1), rstudioapi (>= 0.5), DT(>= 0.17), tibble, dplyr, rio, magrittr, shinyWidgets, lubridate, openxlsx

Description An 'RStudio' addin for editing a 'data.frame' or a 'tibble'. You can delete, add or update a 'data.frame' without coding. You can get resultant data as a 'data.frame'. In the package, modularized 'shiny' app codes are provided. These modules are intended for reuse across applications.

URL https://github.com/cardiomoon/editData

BugReports https://github.com/cardiomoon/editData/issues

License GPL-3

Encoding UTF-8

Depends R (>= 2.10)

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

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

RemoteUrl https://github.com/cardiomoon/editdata

RemoteRef HEAD

RemoteSha 5c539ac7382fdfefd4235e1a7402ba27eb209990

Contents

`checkboxInput3` *Create a side-by-side checkboxInput*

Description

Create a side-by-side checkboxInput

Usage

```
checkboxInput3(inputId, label, value = FALSE, width = 100)
```

Arguments

`inputId`	The input slot that will be used to access the value.
`label`	Display label for the control, or NULL for no label.
`value`	Initial value.
`width`	The width of the input in pixel

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
  ui <- fluidPage(
         label3("Welcome"),
         checkboxInput3("somevalue", "Some value", FALSE),
         verbatimTextOutput("value")
```

```
  )
  server <- function(input, output) {
        output$value <- renderText({ input$somevalue })
  }
  shinyApp(ui, server)
}
```

---

dateInput3 *Create a side-by-side dateInput*

Description

Create a side-by-side dateInput

Usage

```
dateInput3(inputId, label, width = 100, ...)
```

Arguments

inputId	The input slot that will be used to access the value.
label	Display label for the control, or NULL for no label.
width	The width of the input in pixel
...	arguments to be passed to dateInput

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
  ui <- fluidPage(
         label3("Welcome"),
         dateInput3("date", "date"),
         verbatimTextOutput("value")
  )
  server <- function(input, output) {
        output$value <- renderText({ input$date })
  }
  shinyApp(ui, server)
}
```

`editableDT` *Server function of editableDT Shiny module*

Description

Server function of editableDT Shiny module

Usage

```
editableDT(
  input,
  output,
  session,
  data,
  length = 50,
  cols = 1:7,
  status = "default",
  showButtons = TRUE,
  enableSave = TRUE,
  editable = NULL,
  formatList = NULL,
  ...
)
```

Arguments

`input`	input
`output`	output
`session`	session
`data`	A reactive data object
`length`	numeric desired length of string
`cols`	numeric Initial columns to display
`status`	character. dropdownButton status. One of c("default","info","primary","danger","warning","success")
`showButtons`	logical
`enableSave`	logical
`editable`	logical
`formatList`	Null or list. Format list to be passed to formatStyle
`...`	Further arguments to be passed to datatable()

editableDTUI *UI of editableDT Shiny module*

Description

UI of editableDT Shiny module

Usage

```
editableDTUI(id)
```

Arguments

id	A string

Examples

```
# Only run examples in interactive R sessions
if (interactive()) {
library(shiny)
ui=fluidPage(
  selectInput("select","select",choices=c("mtcars","iris","sampleData")),
  textInput("mydata","mydata",value="mtcars"),
  hr(),
  editableDTUI("editableDT"),
  hr(),
  verbatimTextOutput("test")
)
server=function(input,output,session){
  data=reactive({
            myget(input$mydata)
  })
  observeEvent(input$select,{
       updateTextInput(session,"mydata",value=input$select)
  })
  result=callModule(editableDT,"editableDT",data=data)
  output$test=renderPrint({
  str(result())
  })
}
shinyApp(ui=ui,server=server)
}
```

editData *A shiny app for editing a 'data.frame'*

Description

A shiny app for editing a 'data.frame'

Usage

```
editData(
  data = NULL,
  viewer = "dialog",
  length = 50,
  cols = 1:7,
  status = "default",
  showButtons = TRUE,
  enableSave = TRUE,
  editable = NULL,
  formatList = NULL,
  ...
)
```

Arguments

data	A tibble or a tbl_df or a data.frame to manipulate
viewer	Specify where the gadget should be displayed. Possible choices are c("dialog","browser","pane")
length	Numeric desired maximum length of string
cols	numeric
status	character. dropdownButton status. One of c("default","info","primary","danger","warning","success")
showButtons	logical
enableSave	logical
editable	logical
formatList	Null or list. Format list to be passed to formatStyle
...	Further arguments to be passed to datatable()

Value

A manipulated 'data.frame' or NULL

Examples

```
library(shiny)
library(editData)
# Only run examples in interactive R sessions
if (interactive()) {
    result<-editData(mtcars)
    result
}
```

`editFiles` *Edit multiple files side by side*

Description

Edit multiple files side by side

Usage

```
editFiles()
```

`file2ext` *Extract extension from a file name*

Description

Extract extension from a file name

Usage

```
file2ext(filename)
```

Arguments

`filename`	A character string naming a file

label3 *Create a side-by-side label*

Description

Create a side-by-side label

Usage

```
label3(label, width = 100, bg = NULL, ...)
```

Arguments

label	A text to display
width	The width of the input in pixel
bg	The color of text
...	arguments to be passed to label

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
  ui <- fluidPage(
         label3("Welcome"),
         checkboxInput3("somevalue", "Some value", FALSE),
         verbatimTextOutput("value")
  )
  server <- function(input, output) {
        output$value <- renderText({ input$somevalue })
  }
  shinyApp(ui, server)
}
```

makeShort *Truncate string to desired length*

Description

Truncate string to desired length

Usage

```
makeShort(x, length = 50)
```

Arguments

x	A vector
length	numeric desired string length

---

maxLength *Calculate maximal length*

Description

Calculate maximal length

Usage

```
maxLength(x)
```

Arguments

x	A vector

Examples

```
maxLength(month.name)
```

---

myget *Return the Value of a Named data.frame*

Description

Return the Value of a Named data.frame

Usage

```
myget(x)
```

Arguments

x	Name of data.frame

Examples

```
myget("iris")
myget("mtcars")
```

myimport *Read in a data.frame from a file*

Description

Read in a data.frame from a file

Usage

```
myimport(file, ...)
```

Arguments

file	A character string naming a file
...	Further arguments to be passed to rio::import

myimport_csv *read csv file*

Description

read csv file

Usage

```
myimport_csv(file, ...)
```

Arguments

file	A character string naming a file
...	Further arguments to be passed to read.csv

numericInput3 *Create a side-by-side numericInput*

Description

Create a side-by-side numericInput

Usage

```
numericInput3(
  inputId,
  label,
  value,
  min = NA,
  max = NA,
  step = NA,
  width = 100,
  ...
)
```

Arguments

inputId	The input slot that will be used to access the value.
label	Display label for the control, or NULL for no label.
value	Initial value.
min	Minimum allowed value
max	Maximum allowed value
step	Interval to use when stepping between min and max
width	The width of the input in pixel
...	arguments to be passed to numericInput

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
  ui <- fluidPage(
         textInput3("id", "id", ""),
         numericInput3("score","score",value=1)
    )
    server <- function(input, output) {

    }
    shinyApp(ui, server)
}
```

pickerInput3 *Side by side pickerInput*

Description

Side by side pickerInput

Usage

```
pickerInput3(...)
```

Arguments

...	Further arguments to be passed to pickerInput

radioButtons3 *Create a side-by-side radioButtons*

Description

Create a side-by-side radioButtons

Usage

```
radioButtons3(
  inputId,
  label,
  choices,
  bg = NULL,
  labelwidth = 100,
  inline = FALSE,
  align = "right",
  ...
)
```

Arguments

inputId	The input slot that will be used to access the value.
label	Display label for the control, or NULL for no label.
choices	List of values to select from
bg	The color of text
labelwidth	The width of the label in pixel
inline	If TRUE, render the choices inline (i.e. horizontally)
align	text align of label
...	arguments to be passed to radioButtons

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
  ui <- fluidPage(
         label3("Welcome"),
         radioButtons3("mydata", "mydata", choices=c("mtcars","iris")),
         verbatimTextOutput("value")
  )
  server <- function(input, output) {
        output$value <- renderText({ input$mydata })
  }
  shinyApp(ui, server)
}
```

`sampleData` *Sample Data for testing 'editData' addin*

Description

A sample dataset containing data for 4 people

Usage

```
sampleData
```

Format

A data.frame with 4 rows and 6 variables:

name Last name

age age in years

country Country Name

sex sex, A factor with two levels.

bloodType Blood Type. A factor with four levels

date Date

selectInput3 *Create a side-by-side selectInput*

Description

Create a side-by-side selectInput

Usage

```
selectInput3(..., width = 100)
```

Arguments

...	arguments to be passed to selectInput
width	The width of the input in pixel

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
  ui <- fluidPage(
         selectInput3("sex", "sex", choices=c("Male","Female")),
        selectInput3("smoking", "smokingStatus", choices=c("Never","Ex-smoker","Smoker"))
    )
    server <- function(input, output) {

    }
    shinyApp(ui, server)
}
```

selectizeInput3 *side-by-side selectizeInput*

Description

side-by-side selectizeInput

Usage

```
selectizeInput3(..., width = 100)
```

Arguments

...	Further arguments to be passed to selectizeInput
width	Input width in pixel

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
  ui <- fluidPage(
        selectizeInput3("color", "color", choices=colors())
    )
    server <- function(input, output) {

    }
    shinyApp(ui, server)
}
```

`textInput3` *Create a side-by-side textInput control for entry of unstructured text values*

Description

Create a side-by-side textInput control for entry of unstructured text values

Usage

```
textInput3(inputId, label, value = "", width = 100, bg = NULL, ...)
```

Arguments

`inputId`	The input slot that will be used to access the value.
`label`	Display label for the control, or NULL for no label.
`value`	Initial value.
`width`	The width of the input in pixel
`bg`	The color of text
`...`	arguments to be passed to textInput

Examples

```
library(shiny)
# Only run examples in interactive R sessions
if (interactive()) {
  ui <- fluidPage(
        textInput3("id", "id", ""),
        textInput3("name","name","")
    )
    server <- function(input, output) {

    }
    shinyApp(ui, server)
}
```

Index