Package 'shinyStorePlus'

November 21, 2022

Type Package

Title Secure in-Browser Storage for 'Shiny' Inputs and Variables

Version 0.8

Maintainer Obinna Obianom <idonshayo@gmail.com>

Description

Store persistent and synchronized data from 'Shiny' inputs within the browser in a secure format. Refresh 'Shiny' applications and preserve user-inputs over multiple sessions. A database-like storage format is implemented using 'Dexie.js' <https://dexie.org>, a minimal wrapper for 'IndexedDB'. Transfer browser link parameters to 'Shiny' input or output values.

License MIT + file LICENSE

URL https://github.com/oobianom/shinyStorePlus

BugReports https://github.com/oobianom/shinyStorePlus

Depends R (>= 3.6)

Imports shiny, jsonlite, utils, htmltools

Suggests rmarkdown, knitr, qpdf

Encoding UTF-8

VignetteBuilder knitr

Language en-US

LazyData false

RoxygenNote 7.2.1

NeedsCompilation no

Author Obinna Obianom [aut, cre]

Repository CRAN

Date/Publication 2022-11-21 07:50:02 UTC

R topics documented:

clearStore 2
initStore 3
link2input 4
seeexample 5
setupStorage 6

Index **10**

`clearStore` *Clear storage for an application*

Description

Remove all stored inputs of an application

Usage

```
clearStore(appId, session = getDefaultReactiveDomain())
```

Arguments

`appId`	the application identification
`session`	session object

Value

No return value, called for side effects

Note

Ensure not to use this function when the inputs are intended to be tracked.

Examples

```
if (interactive()) {
  library(shiny)
  library(shinyStorePlus)

  ui <- fluidPage(
    # initialize stores
    initStore(),
    "Sample delete storage",
    selectInput("dataset",
      label = "Dataset",
      choices = c("dataset 1", "dataset 2")
    )
  )
  server <- function(input, output, session) {
```

```
    appid <- "application01"
    clearStore(appId = appid)
  }
  shinyApp(ui, server)
}
```

---

initStore *Included package scripts*

Description

Include Dexie and the package script in the header

Usage

```
initStore()
```

Value

Initialize the storage by including scripts necessary for the persistent storage handling

Examples

```
library(shiny)
library(shinyStorePlus)

if (interactive()) {
  ui <- shiny::fluidPage(
    # initialize stores
    initStore(),
    titlePanel("Sample
             shinyStorePlus Init Inputs"),
    sidebarLayout(
      sidebarPanel(
        sliderInput("nextgenshinyapps1",
          "Number of bins:",
          min = 1,
          max = 200,
          value = 150
        ),
        textInput(
          "caption",
          "simple caption:",
          "summary, try editing"
        ),
        numericInput("obs",
          "sample observations:",
          10,
          min = 1, max = 100
```

```
        )
      ),
      mainPanel(
        plotOutput("distPlot")
      )
    )
  )
  server <- function(input, output, session) {
    output$distPlot <- renderPlot({
      x <- faithful[, 2]
      bins <- seq(min(x),
        max(x),
        length.out =
          input$nextgenshinyapps1 + 1
      )
      hist(x,
        breaks = bins,
        col = "blue",
        border = "gray"
      )
    })
  }
  shiny::shinyApp(ui = ui, server = server)
}
```

`link2input` *Convert Browser Location Parameters to Shiny Input and Output Values*

Description

Parse the browser link and retrieve parameters for inclusion as values in inputs or outputs

Usage

```
link2input(..., inputtype = "default")
```

Arguments

`...`	List of Shiny input IDs to match with window location parameters
`inputtype`	Type of inputs being included

Value

Setting of the Shiny inputs to the values of the parameters in the browser link

Note

a great example of how to use this functionality can be found in https://cran.r-project.org/web/packages/shinyStorePlus/vignet

Examples

```
if (interactive()) {
  # within the server function
  server <- function(input, output, session) {
    link2input(
      cd323 = "name",
      datasetbin = "data",
      numberid = "num"
    )

    link2input(
      outputid = "outt",
      inputtype = "output"
    )
  }
}
```

`seeexample` *Load the example for the package*

Description

Example of a shiny application with secure in-browser storage of inputs when changed

Usage

```
seeexample(name = c("storeInputs", "browserLinkToInput"))
```

Arguments

`name`	the name of example to view. choices include storeInputs or browserLinkToInput

Value

An example of inputs persistently stored when changed and the page refreshed

Note

Changes made to the input will be saved and returned when the page is refresh within the same browser over different sessions

Examples

```
if (interactive()) {
  seeexample()
  seeexample("browserLinkToInput")
}
```

`setupStorage` *Set up inputs for storage*

Description

Set up the application and inputs to track and retrieve stores

Usage

```
setupStorage(appId, inputs = TRUE)
```

Arguments

`appId`	your desired application id
`inputs`	choose whether to track all inputs or specific input variables

Value

Embed within a page storage that allows input changes to be saved without slowing down the shiny application

Note

the inputs argument may be a `TRUE` or `FALSE` or a list of input ids

Examples

```
library(shiny)
library(shinyStorePlus)

# example 1 that tracks all inputs
if (interactive()) {
  ui <- shiny::fluidPage(
    titlePanel("EX1
               shinyStorePlus All Inputs"),
    initStore(),
    sidebarLayout(
      sidebarPanel(
        sliderInput("nextgenshinyapps1",
```

```
            "Number of bins:",
            min = 1,
            max = 200,
            value = 150
          ),
          textInput(
            "caption",
            "simple caption:",
            "try editing - r2resize pkg"
          ),
          numericInput("obs",
            "sample observations:",
            10,
            min = 1, max = 100
          )
        ),
        mainPanel(
          plotOutput("distPlot")
        )
      )
    )
    server <- function(input, output, session) {
      output$distPlot <- renderPlot({
        x <- faithful[, 2]
        bins <- seq(min(x),
          max(x),
          length.out =
            input$nextgenshinyapps1 + 1
        )
        hist(x,
          breaks = bins,
          col = "blue",
          border = "gray"
        )
      })

      # insert at the bottom
      appid <- "application01"
      setupStorage(
        appId = appid,
        inputs = TRUE
      )
    }
    shiny::shinyApp(ui = ui, server = server)
  }


  # example 2 that tracks only 2 inputs
  if (interactive()) {
    ui <- shiny::fluidPage(
      # init stores
      initStore(),
      titlePanel("Ex2:
```

```
            shinyStorePlus Some Inputs"),
    sidebarLayout(
      sidebarPanel(
        sliderInput("nextgenshinyapps1",
          "Number of bins:",
          min = 1,
          max = 200,
          value = 150
        ),
        textInput(
          "caption",
          "simple caption:",
          "summary, try editing"
        ),
        numericInput("obs",
          "sample observations:",
          10,
          min = 1, max = 100
        )
      ),
      mainPanel(
        plotOutput("distPlot")
      )
    )
  )
  server <- function(input, output, session) {
    output$distPlot <- renderPlot({
      x <- faithful[, 2]
      bins <- seq(min(x),
        max(x),
        length.out =
          input$nextgenshinyapps1 + 1
      )
      hist(x,
        breaks = bins,
        col = "blue",
        border = "gray"
      )
    })

    # insert at the bottom  !!!IMPORTANT
    appid <- "application023"
    setupStorage(
      appId = appid,
      inputs = list(
        "nextgenshinyapps1",
        "caption"
      )
    )
  }
  shiny::shinyApp(ui = ui, server = server)
}
```

Index

clearStore, 2

initStore, 3

link2input, 4

seeexample, 5
setupStorage, 6