Package 'shiny.destroy'

May 9, 2026

Type Package

Title Create Destroyable Modules in 'Shiny'

Version 0.1.0

Description Enables the complete removal of various 'Shiny' components, such as inputs, outputs and modules. It also aids in the removal of observers that have been created in dynamically created modules.

License MIT + file LICENSE

Encoding UTF-8

Imports purrr, rlang, shiny

Suggests bslib, htmltools, knitr, rmarkdown, shinytest2, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Language en-GB

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Ashley Baldry [aut, cre]

Maintainer Ashley Baldry <arbaldry91@gmail.com>

Repository CRAN

Date/Publication 2024-09-17 12:30:02 UTC

Contents

`shiny.destroy-package` *shiny.destroy: Create Destroyable Modules in 'Shiny'*

Description

This package enables the complete removal of various 'Shiny' components, such as inputs, outputs and modules. It also aids in the removal of observers that have been created in dynamically created modules.

Author(s)

Maintainer: Ashley Baldry <arbaldry91@gmail.com>

`addDestroyers` *Add shiny.destroy Code to Module*

Description

For a given 'moduleServer' call, add the code required for {shiny.destroy} to work. This will involve creating the observer list to the user session, and adds all observers within the list.

Usage

```
addDestroyers(module)
```

Arguments

`module`	The function call to 'moduleServer'

Value

An updated version of 'module', where the {shiny.destroy} code has been added.

`destroyModule` *Destroy Shiny Module*

Description

Given the namespace of a shiny module, remove all references to the inputs, outputs and observers that are called within the module.

Usage

```
destroyModule(id = NULL, session = getDefaultReactiveDomain())
```

Arguments

`id`	The module namespace ID. Use 'NULL' to destroy the module the call is being executed in.
`session`	The shiny session, by default it is 'shiny::getDefaultReactiveDomain()'

Value

No return value, called to remove the relevant module UI and server-side observers.

Examples

```
library(shiny)

basicModuleUI <- function(id) {
  ns <- NS(id)
  actionButton(ns("click"), "Click Button")
}

basicModuleServer <- function(id) {
  moduleServer(id, function(input, output, session) {
    rv <- reactiveVal(0L)
    observeEvent(input$click, rv(rv() + 1L))
    rv
  })
}

destroyableModuleUI <- makeModuleUIDestroyable(basicModuleUI)
destroyableModuleServer <- makeModuleServerDestroyable(basicModuleServer)

ui <- fluidPage(
  destroyableModuleUI(id = "test"),
  actionButton("destroy", "Destroy module"),
  textOutput("reactive_value")
)

server <- function(input, output, session) {
```

```
  top_rv <- reactiveVal()
  reactive_value <- destroyableModuleServer("test")
  observeEvent(reactive_value(), top_rv(reactive_value()))

  output$reactive_value <- renderText(top_rv())

  observeEvent(input$destroy, destroyModule("test"))
}

shinyApp(ui, server)
```

---

isSpecifiedFunction *Check if Function Call is relevant function*

Description

A short description...

Usage

```
isSpecifiedFunction(fn_call, fns)
```

Arguments

fn_call	A function call
fns	A character vector of functions to compare the function call against

Value

A logical value stating whether or not the function call is in the collection.

---

makeModuleUIDestroyable *Create Destroyable Module*

Description

Adding wrappers to a shiny module to enable an ease of dynamically adding and removing modules within a shiny application.

Usage

```
makeModuleUIDestroyable(module_fn, wrapper = shiny::div)

makeModuleServerDestroyable(module_fn)
```

Arguments

module_fn	The server-side part of the module
wrapper	If the module is a ‘shiny::tagList()‘, then an HTML tag will be wrapped by an HTML tag so that a shiny.destroy attribute can be attached

Value

An updated function call of ‘module_fn‘.

For the UI, if the returned object from ‘module_fn‘ is a ‘shiny.tag‘ then an additional attribute will be added to the top-level HTML tag for {shiny.destroy} to reference when removing the UI. If the returned object is a ‘shiny.tag.list‘ then a wrapper tag will surround the module with the attribute to destroy the module.

For the server, each observer will be assigned to the ‘.shiny.destroy‘ list within ‘session$userData‘. The returned object from the module remains the same as before.

Examples

```
library(shiny)

# UI
basicModuleUI <- function(id) {
  ns <- NS(id)
  actionButton("click", "Increase")
}

destroyableModuleUI <- makeModuleUIDestroyable(basicModuleUI)

# Server-side
basicMoudleServer <- function(id) {
  moduleServer(id, function(input, output, session) {
    rv <- reactiveVal()
    observeEvent(input$click, rv(input$click))
  })
}

destroyableModuleServer <- makeModuleServerDestroyable(basicMoudleServer)

# Shiny Application
ui <- fluidPage(
  destroyableModuleUI(id = "test"),
  actionButton("destroy", "Destroy module"),
  textOutput("reactive_value")
)

server <- function(input, output, session) {
  top_rv <- reactiveVal()

  reactive_value <- destroyableModuleServer("test")
  observeEvent(reactive_value(), top_rv(reactive_value()))
```

```
  output$reactive_value <- renderText(top_rv())

  observeEvent(input$destroy, destroyModule("test"))
}
```

`removeInput` *Remove Input from Shiny Session*

Description

The removal of the named input in a shiny session.

Usage

```
removeInput(
  id,
  selector = paste0("#", id),
  session = getDefaultReactiveDomain()
)
```

Arguments

`id`	Input value name
`selector`	The HTML selector to remove the UI for. By default it is the tag where the ID matches the input, but might need to be adjusted for different inputs.
`session`	The Shiny session to remove the input from

Details

If the input is a standard shiny input e.g. ‘numericInput‘, then to remove the label as well as the input, set the selector to be ‘paste0(":has(> #", id, ")")‘

Value

An invisible ‘TRUE‘ value confirming that the input has been removed.

Examples

```
library(shiny)
library(shiny.destroy)

ui <- fluidPage(
  numericInput("number", "Select number:", 5, 1, 10),
  p("Selected number:", textOutput("number_out", inline = TRUE)),
  actionButton("delete", "Remove input")
)
```

```
server <- function(input, output, session) {
  output$number_out <- renderText(input$number %||% "input unavailable")

  observeEvent(
    input$delete,
    removeInput("number", selector = ":has(> #number)")
   )
}

shinyApp(ui, server)
```

`removeOutput` *Remove Output from Shiny Session*

Description

The removal of the named output in a shiny session.

Usage

```
removeOutput(
  id,
  selector = paste0("#", id),
  session = getDefaultReactiveDomain()
)
```

Arguments

`id`	Output value name
`selector`	The HTML selector to remove the UI for. By default it is the tag where the ID matches the output, but might need to be adjusted for different inputs.
`session`	The Shiny session to remove the output from

Value

An invisible 'TRUE' value confirming that the output has been removed.

Examples

```
library(shiny)
library(shiny.destroy)

ui <- fluidPage(
  numericInput("number", "Select number:", 5, 1, 10),
  p("Selected number:", textOutput("number_out", inline = TRUE)),
  actionButton("delete", "Remove output")
)
```

```
server <- function(input, output, session) {
  output$number_out <- renderText(input$number)

  observeEvent(
    input$delete,
    removeOutput("number_out")
   )
}

shinyApp(ui, server)
```

`runDestroyExample` *Run 'shiny.destroy' example*

Description

To see how the 'shiny.destroy' works, examples are provided within the package.

Usage

```
runDestroyExample(example = NA, ...)
```

Arguments

`example`	The name of the example to run, or `NA` (the default) to list the available examples.
`...`	Additional parameters sent to 'shiny::runExample'

Details

The following examples are available:

01_boxes A simple application where the "create" button will load a simple box with a "destroy" button. This highlights the full removal of the module when the button is pressed.

02_sleep An application that has 2 side by side modules, one using {shiny} to remove the UI and the other using {shiny.destroy} to fully remove the boxes to display the incremental time gain from removing the long-running observers.

Value

The shiny application displayed in the specified location.

Examples

```
runDestroyExample("01_boxes")
```

Index