Package 'shinystate'

September 18, 2025

Title Customization of Shiny Bookmarkable State

Version 0.1.0

Description Enhance the bookmarkable state feature of 'shiny' with additional customization such as storage location and storage repositories leveraging the 'pins' package.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.0.0)

Imports archive, fs, htmltools, pins, R6, shiny (>= 0.14),

Suggests bslib, DT, knitr, lubridate, rlang, rmarkdown, roxy.shinylive, testthat (>= 3.0.0), withr

Config/testthat/edition 3

URL https://rpodcast.github.io/shinystate/, https://github.com/rpodcast/shinystate

BugReports https://github.com/rpodcast/shinystate/issues

VignetteBuilder knitr

NeedsCompilation no

Author Eric Nantz [aut, cre] (ORCID: <https://orcid.org/0000-0001-8104-7510>), Eli Lilly and Company [cph, fnd]

Maintainer Eric Nantz <theRcast@gmail.com>

Repository CRAN

Date/Publication 2025-09-18 08:10:01 UTC

Contents

`StorageClass` *StorageClass R6 class*

Description

This class provides a set of methods to create and manage Shiny bookmarkable state files.

Public fields

`local_storage_dir` file path to use for storing bookmarkable state files. If not specified, a temporary directory on the host system will be used.

`board_sessions` Optional pre-created board object created with the pins package. If missing, a folder-based pin board will be created using the `local_storage_dir` path.

Methods

Public methods:

- `StorageClass$new()`
- `StorageClass$get_sessions()`
- `StorageClass$restore()`
- `StorageClass$snapshot()`
- `StorageClass$delete()`
- `StorageClass$register_metadata()`
- `StorageClass$clone()`

Method `new()`**:** Initialize a `StorageClass` object

Usage:

```
StorageClass$new(local_storage_dir = NULL, board_sessions = NULL)
```

Arguments:

`local_storage_dir` file path to use for storing bookmarkable state files. If not specified, a temporary directory on the host system will be used.

`board_sessions` Optional pre-created board object created with the pins package. If missing, a folder-based pin board will be created using the `local_storage_dir` path.

Returns: An object with class `StorageClass` and the methods described in this documentation

Examples:

```
## Only run examples in interactive R sessions
if (interactive()) {

# beginning of application
library(shiny)
library(shinystate)

# Create a StorageClass object with default settings
```

```
storage <- StorageClass$new()

# Use a pre-specified directory to store state files
# For purposes of this example, use a temporary directory
storage <- StorageClass$new(local_storage_dir = tempdir())

# use a custom pins board to store bookmarkable state data
# For purposes of this example, use a temporary directory
library(pins)
board <- board_temp()
storage <- StorageClass$new(board_sessions = board)
}
```

Method `get_sessions()`**:** Obtain saved bookmarkable state session metadata

Calls `$get_sessions()` on the `StorageClass` object to extract the bookmarkable state session metadata. You can leverage this data frame in your Shiny application to let the user manage their existing bookmarkable state sessions, for example.

Usage:

```
StorageClass$get_sessions()
```

Returns: data frame of bookmarkable session metadata if at least one bookmarkable state session has been saved. Otherwise, the return object will be `NULL`.

Examples:

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
library(shinystate)

# Create a StorageClass object with default settings
storage <- StorageClass$new()

# obtain session data
storage$get_sessions()
}
```

Method `restore()`**:** Restore a previous bookmarkable state session

Usage:

```
StorageClass$restore(url, session = shiny::getDefaultReactiveDomain())
```

Arguments:

`url` character with the unique URL assigned to the bookmarkable state session.

`session` The Shiny session to associate with the restore operation

Examples:

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shinystate)
```

```
# Create a StorageClass object with default settings
storage <- StorageClass$new()

# obtain session data
session_df <- storage$get_sessions()

# restore state
# typically run inside a shiny observe or observeEvent call
storage$restore(tail(session_df$url, n = 1))
}
```

Method `snapshot()`**:** Create a snapshot of bookmarkable state

Usage:

```
StorageClass$snapshot(
  session_metadata = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`session_metadata` Optional named list of additional variables to include with the default bookmarkable state attributes when creating the snapshot. Each element of the list must be a single-length item

`session` The Shiny session to associate with the snapshot operation

Examples:

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shinystate)

# Create a StorageClass object with default settings
storage <- StorageClass$new()

# save state with timestamp as metadata
# typically run inside a shiny observe or observeEvent call
storage$snapshot(session_metadata = list(time = Sys.time()))
}
```

Method `delete()`**:** Delete a previous snapshot of bookmarkable state

Usage:

```
StorageClass$delete(url)
```

Arguments:

`url` character with the unique URL assigned to the bookmarkable state session.

Examples:

```
## Only run examples in interactive R sessions
if (interactive()) {
```

```
library(shinystate)

# Create a StorageClass object with default settings
storage <- StorageClass$new()

# obtain session data
session_df <- storage$get_sessions()

# delete a session
# typically run inside a shiny observe or observeEvent call
storage$delete(session_df$url[1])
}
```

Method `register_metadata()`**:** Register bookmarkable state storage data collection

This method must be called in the application server function to perform the necessary customizations to bookmarkable state methods. This function is meant to be called near the beginning of the Shiny application server function.

Usage:

```
StorageClass$register_metadata()
```

Examples:

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shinystate)

# Create a StorageClass object with default settings
storage <- StorageClass$new()

# application server code
server <- function(input, output, session) {
  storage$register_metadata()
}
}
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
StorageClass$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `StorageClass$new`
## ------------------------------------------------
```

```
## Only run examples in interactive R sessions
if (interactive()) {

# beginning of application
library(shiny)
library(shinystate)

# Create a StorageClass object with default settings
storage <- StorageClass$new()

# Use a pre-specified directory to store state files
# For purposes of this example, use a temporary directory
storage <- StorageClass$new(local_storage_dir = tempdir())

# use a custom pins board to store bookmarkable state data
# For purposes of this example, use a temporary directory
library(pins)
board <- board_temp()
storage <- StorageClass$new(board_sessions = board)
}

## ------------------------------------------------
## Method `StorageClass$get_sessions`
## ------------------------------------------------

## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
library(shinystate)

# Create a StorageClass object with default settings
storage <- StorageClass$new()

# obtain session data
storage$get_sessions()
}

## ------------------------------------------------
## Method `StorageClass$restore`
## ------------------------------------------------

## Only run examples in interactive R sessions
if (interactive()) {

library(shinystate)

# Create a StorageClass object with default settings
storage <- StorageClass$new()

# obtain session data
session_df <- storage$get_sessions()
```

```
# restore state
# typically run inside a shiny observe or observeEvent call
storage$restore(tail(session_df$url, n = 1))
}

## ------------------------------------------------
## Method `StorageClass$snapshot`
## ------------------------------------------------

## Only run examples in interactive R sessions
if (interactive()) {

library(shinystate)

# Create a StorageClass object with default settings
storage <- StorageClass$new()

# save state with timestamp as metadata
# typically run inside a shiny observe or observeEvent call
storage$snapshot(session_metadata = list(time = Sys.time()))
}

## ------------------------------------------------
## Method `StorageClass$delete`
## ------------------------------------------------

## Only run examples in interactive R sessions
if (interactive()) {

library(shinystate)

# Create a StorageClass object with default settings
storage <- StorageClass$new()

# obtain session data
session_df <- storage$get_sessions()

# delete a session
# typically run inside a shiny observe or observeEvent call
storage$delete(session_df$url[1])
}

## ------------------------------------------------
## Method `StorageClass$register_metadata`
## ------------------------------------------------

## Only run examples in interactive R sessions
if (interactive()) {

library(shinystate)

# Create a StorageClass object with default settings
storage <- StorageClass$new()
```

```
# application server code
server <- function(input, output, session) {
  storage$register_metadata()
}
}
```

`use_shinystate` *Add shinystate dependency*

Description

Include shinystate dependencies in your Shiny application UI

Usage

```
use_shinystate()
```

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
library(shinystate)

storage <- StorageClass$new()

ui <- function(request) {
  fluidPage(
    use_shinystate(),
    actionButton("bookmark", "Bookmark"),
    actionButton("restore", "Restore Last Bookmark")
  )
}
}
```

Index