Package ‘shinyauthr’

July 20, 2021

Type Package

Title 'Shiny' Authentication Modules

Version 1.0.0

Description Add in-app user authentication to 'shiny', allowing you to secure publicly hosted apps and build dynamic user interfaces from user information.

License MIT + file LICENSE

Encoding UTF-8

Imports shiny (>= 1.5.0), shinyjs, dplyr, rlang, sodium, glue

Suggests DBI, RSQLite, lubridate, shinydashboard, testthat (>= 3.0.0), shineshinytest, knitr, rmarkdown, covr

RoxygenNote 7.1.1

URL https://github.com/paulc91/shinyauthr

BugReports https://github.com/paulc91/shinyauthr/issues

Config/testthat/edition 3

NeedsCompilation no

Author Paul Campbell [aut, cre] (<https://orcid.org/0000-0003-1018-6606>), Michael Dewar [ctb]

Maintainer Paul Campbell <pacampbell91@gmail.com>

Repository CRAN

Date/Publication 2021-07-20 07:20:02 UTC

R topics documented:

login ................................................................. 2
loginServer ...................................................... 3
loginUI ............................................................ 5
logout ............................................................. 7
logoutServer ..................................................... 8
logoutUI .......................................................... 9
runExample ....................................................... 11
login

login server module (deprecated)

Description

Deprecated. Use \texttt{loginServer} instead.

Arguments

\begin{itemize}
\item \texttt{input} \quad shiny input
\item \texttt{output} \quad shiny output
\item \texttt{session} \quad shiny session
\item \texttt{data} \quad data frame or tibble containing usernames, passwords and other user data
\item \texttt{user\_col} \quad bare (unquoted) column name containing usernames
\item \texttt{pwd\_col} \quad bare (unquoted) column name containing passwords
\item \texttt{sodium\_hashed} \quad have the passwords been hash encrypted using the sodium package? defaults to \texttt{FALSE}
\item \texttt{hashed} \quad Deprecated. \texttt{shinyauthr} now uses the sodium package for password hashing and decryption. If you have previously hashed your passwords with the digest package to use with \texttt{shinyauthr} please re-hash them with sodium for decryption to work.
\item \texttt{algo} \quad Deprecated
\item \texttt{log\_out} \quad [reactive] supply the returned reactive from \texttt{logout} here to trigger a user logout
\item \texttt{sessionid\_col} \quad bare (unquoted) column name containing session ids
\item \texttt{cookie\_getter} \quad a function that returns a data.frame with at least two columns: user and session
\item \texttt{cookie\_setter} \quad a function with two parameters: user and session. The function must save these to a database.
\item \texttt{reload\_on\_logout} \quad should app force reload on logout?
\end{itemize}

Details

Shiny authentication module for use with \texttt{loginUI}
Call via \texttt{shiny::callModule(shinyauthr::login,"id",...)}

This function is now deprecated in favour of \texttt{loginServer} which uses shiny’s new \texttt{moduleServer} method as opposed to the \texttt{callModule} method used by this function. See the \texttt{loginServer} documentation for details on how to migrate.

Value

The module will return a reactive 2 element list to your main application. First element \texttt{user\_auth} is a boolean indicating whether there has been a successful login or not. Second element \texttt{info} will be the data frame provided to the function, filtered to the row matching the successfully logged in username. When \texttt{user\_auth} is \texttt{FALSE} \texttt{info} is \texttt{NULL}. 

loginServer

Examples

```r
## Not run:
user_credentials <- shiny::callModule(
  login,
  id = "login",
  data = user_base,
  user_col = user,
  pwd_col = password,
  log_out = reactive(logout_init())
)

## End(Not run)
```

---

loginServer  

**login server module**

Description

Shiny authentication module for use with `loginUI`

Usage

```r
loginServer(
  id,
  data,
  user_col,
  pwd_col,
  sodium_hashed = FALSE,
  log_out = shiny::reactiveVal(),
  reload_on_logout = FALSE,
  cookie_logins = FALSE,
  sessionid_col,
  cookie_getter,
  cookie_setter
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>An ID string that corresponds with the ID used to call the module’s UI function</td>
</tr>
<tr>
<td>data</td>
<td>data frame or tibble containing user names, passwords and other user data</td>
</tr>
<tr>
<td>user_col</td>
<td>bare (unquoted) or quoted column name containing user names</td>
</tr>
<tr>
<td>pwd_col</td>
<td>bare (unquoted) or quoted column name containing passwords</td>
</tr>
<tr>
<td>sodium_hashed</td>
<td>have the passwords been hash encrypted using the sodium package? defaults to FALSE</td>
</tr>
</tbody>
</table>
loginServer

log_out [reactive] supply the returned reactive from logoutServer here to trigger a user logout

reload_on_logout
should app force a session reload on logout?

cookie_logins enable automatic logins via browser cookies?

sessionid_col bare (unquoted) or quoted column name containing session ids

cookie_getter a function that returns a data.frame with at least two columns: user and session

cookie_setter a function with two parameters: user and session. The function must save these to a database.

Details
This module uses shiny’s new moduleServer method as opposed to the callModule method used by the now deprecated login function and must be called differently in your app. For details on how to migrate see the ‘Migrating from callModule to moduleServer’ section of Modularizing Shiny app code.

Value
The module will return a reactive 2 element list to your main application. First element user_auth is a boolean indicating whether there has been a successful login or not. Second element info will be the data frame provided to the function, filtered to the row matching the successfully logged in username. When user_auth is FALSE info is NULL.

Examples
library(shiny)

# dataframe that holds usernames, passwords and other user data
user_base <- dplyr::tibble(
  user = c("user1", "user2"),
  password = c("pass1", "pass2"),
  permissions = c("admin", "standard"),
  name = c("User One", "User Two")
)

ui <- fluidPage(
  # add logout button UI
  div(class = "pull-right", shinyauthr::logoutUI(id = "logout")),
  # add login panel UI function
  shinyauthr::loginUI(id = "login"),
  # setup table output to show user info after login
  tableOutput("user_table")
)

server <- function(input, output, session) {
  # call login module supplying data frame,
  # user and password cols and reactive trigger
  credentials <- shinyauthr::loginServer(id = "login",
    # data frame that holds usernames, passwords and other user data
    user_base <- dplyr::tibble(
      user = c("user1", "user2"),
      password = c("pass1", "pass2"),
      permissions = c("admin", "standard"),
      name = c("User One", "User Two")
    )
  )
  # setup table output to show user info after login
  tableOutput("user_table")
  # add login button UI
  div(class = "pull-right", shinyauthr::logoutUI(id = "logout")),
  # add login panel UI function
  shinyauthr::loginUI(id = "login"),
  # setup table output to show user info after login
  tableOutput("user_table")
}
```r
data = user_base,
user_col = user,
pwd_col = password,
log_out = reactive(logout_init())
)

# call the logout module with reactive trigger to hide/show
logout_init <- shinyauthr::logoutServer(
  id = "logout",
  active = reactive(creds()$user_auth)
)

output$user_table <- renderTable({
  # use req to only render results when credentials()$user_auth is TRUE
  req(creds()$user_auth)
  creds()$info
})
}

if (interactive()) shinyApp(ui = ui, server = server)
```

---

**loginUI**

*login UI module*

### Description

Shiny UI Module for use with `loginServer`

### Usage

```r
loginUI(
  id,
  title = "Please log in",
  user_title = "User Name",
  pass_title = "Password",
  login_title = "Log in",
  error_message = "Invalid username or password!",
  additional_ui = NULL,
  cookie_expiry = 7
)
```

### Arguments

- **id**
  - An ID string that corresponds with the ID used to call the module’s server function
- **title**
  - header title for the login panel
- **user_title**
  - label for the user name text input
- **pass_title**
  - label for the password text input
login_title label for the login button
error_message message to display after failed login
additional_ui additional shiny UI element(s) to add below login button. Wrap multiple inside `shiny::tagList()`
cookie_expiry number of days to request browser to retain login cookie

Value

Shiny UI login panel with user name text input, password text input and login action button.

Examples

```r
library(shiny)

# dataframe that holds usernames, passwords and other user data
user_base <- dplyr::tibble(
  user = c("user1", "user2"),
  password = c("pass1", "pass2"),
  permissions = c("admin", "standard"),
  name = c("User One", "User Two")
)

ui <- fluidPage(
  # add logout button UI
  div(class = "pull-right", shinyauthr::logoutUI(id = "logout")),
  # add login panel UI function
  shinyauthr::loginUI(id = "login"),
  # setup table output to show user info after login
  tableOutput("user_table")
)

server <- function(input, output, session) {
  # call login module supplying data frame, user and password cols and reactive trigger
  credentials <- shinyauthr::loginServer(
    id = "login",
    data = user_base,
    user_col = user,
    pwd_col = password,
    log_out = reactive(logout_init())
  )

  # call the logout module with reactive trigger to hide/show
  logout_init <- shinyauthr::logoutServer(
    id = "logout",
    active = reactive(credentials()$user_auth)
  )

  output$user_table <- renderTable({
    # use req to only render results when credentials()$user_auth is TRUE
    req(credentials()$user_auth)
  })
}
```
logout

```r
credentials()$info
}
}

if (interactive()) shinyApp(ui = ui, server = server)
```

---

**Description**

Deprecated. Use `logoutServer` instead.

**Arguments**

- `input` shiny input
- `output` shiny output
- `session` shiny session
- `active` [reactive] supply the returned user_auth boolean reactive from `login` here to hide/show the logout button

**Details**

Shiny authentication module for use with `logoutUI`

Call via `shiny::callModule(shinyauthr::logout,"id",...)`

This function is now deprecated in favour of `logoutServer` which uses shiny’s new `moduleServer` method as opposed to the `callModule` method used by this function. See the `logoutServer` documentation For details on how to migrate.

**Value**

Reactive boolean, to be supplied as the `log_out` argument of the `login` module to trigger the logout process

**Examples**

```r
## Not run:
logout_init <- shiny::callModule(
  logout,
  id = "logout",
  active = reactive(user_credentials()$user_auth)
)

## End(Not run)
```
logoutServer

logout server module

**Description**

Shiny authentication module for use with `logoutUI`

**Usage**

```r
logoutServer(id, active, ...)
```

**Arguments**

- `id` An ID string that corresponds with the ID used to call the module’s UI function
- `active` reactive supply the returned `user_auth` boolean reactive from `loginServer` here to hide/show the logout button
- `...` arguments passed to `toggle`

**Details**

This module uses shiny’s new `moduleServer` method as opposed to the `callModule` method used by the now deprecated `login` function and must be called differently in your app. For details on how to migrate see the ‘Migrating from callModule to moduleServer’ section of Modularizing Shiny app code.

**Value**

Reactive boolean, to be supplied as the `log_out` argument of the `loginServer` module to trigger the logout process

**Examples**

```r
library(shiny)

# dataframe that holds usernames, passwords and other user data
user_base <- dplyr::tibble(  
  user = c("user1", "user2"),
  password = c("pass1", "pass2"),
  permissions = c("admin", "standard"),
  name = c("User One", "User Two")
)

ui <- fluidPage(  
  # add logout button UI
  div(class = "pull-right", shinyauthr::logoutUI(id = "logout")),
  # add login panel UI function
  shinyauthr::loginUI(id = "login"),
  # setup table output to show user info after login
```
server <- function(input, output, session) {
  # call login module supplying data frame,
  # user and password cols and reactive trigger
  credentials <- shinyauthr::loginServer(
    id = "login",
    data = user_base,
    user_col = user,
    pwd_col = password,
    log_out = reactive(logout_init())
  )

  # call the logout module with reactive trigger to hide/show
  logout_init <- shinyauthr::logoutServer(
    id = "logout",
    active = reactive(credentials()$user_auth)
  )

  output$user_table <- renderTable({
    # use req to only render results when credentials()$user_auth is TRUE
    req(credentials()$user_auth)
    credentials()$info
  })
}

if (interactive()) shinyApp(ui = ui, server = server)

---

**logoutUI**  
**logout UI module**

**Description**

Shiny UI Module for use with `logoutServer`

**Usage**

```
logoutUI(
  id,
  label = "Log out",
  icon = NULL,
  class = "btn-danger",
  style = "color: white;"
)
```

**Arguments**

- `id`  
  An ID string that corresponds with the ID used to call the module’s server function
label    label for the logout button
icon     An optional icon to appear on the button.
class    bootstrap class for the logout button
style    css styling for the logout button

Value
Shiny UI action button

Examples

```r
library(shiny)

# dataframe that holds usernames, passwords and other user data
user_base <- dplyr::tibble(
  user = c("user1", "user2"),
  password = c("pass1", "pass2"),
  permissions = c("admin", "standard"),
  name = c("User One", "User Two")
)

ui <- fluidPage(
  # add logout button UI
  div(class = "pull-right", shinyauthr::logoutUI(id = "logout")),
  # add login panel UI function
  shinyauthr::loginUI(id = "login"),
  # setup table output to show user info after login
  tableOutput("user_table")
)

server <- function(input, output, session) {
  # call login module supplying data frame,
  # user and password cols and reactive trigger
  credentials <- shinyauthr::loginServer(
    id = "login",
    data = user_base,
    user_col = user,
    pwd_col = password,
    log_out = reactive(logout_init())
  )

  # call the logout module with reactive trigger to hide/show
  logout_init <- shinyauthr::logoutServer(
    id = "logout",
    active = reactive(credentials()$user_auth)
  )

  # use req to only render results when credentials()$user_auth is TRUE
  output$user_table <- renderTable(
    req(credentials()$user_auth)
    credentials()$info
  )
}
```
runExample

if (interactive()) shinyApp(ui = ui, server = server)

runExample  Run shinyauthr examples

Description

Launch an example shiny app using shinyauthr authentication modules. Use user1 pass1 or user2 pass2 to login.

Usage

runExample(example = c("basic", "shinydashboard", "navbarPage"))

Arguments

dexample  The app to launch. Options are "basic", "shinydashboard" or "navbarPage"

Value

No return value, a shiny app is launched.

Examples

## Only run this example in interactive R sessions
if (interactive()) {
  runExample("basic")
  runExample("shinydashboard")
  runExample("navbarPage")
}
Index

callModule, 2, 4, 7, 8

icon, 10

login, 2, 4, 7, 8
loginServer, 2, 3, 5, 8
loginUI, 2, 3, 5
logout, 2, 7
logoutServer, 4, 7, 8, 9
logoutUI, 7, 8, 9

moduleServer, 2, 4, 7, 8

runExample, 11

toggle, 8