Package ‘login’

Type Package
Title 'shiny' Login Module
Version 0.9.3
Date 2024-04-17
Maintainer Jason Bryer <jason@bryer.org>
Description Framework for adding authentication to 'shiny' applications. Provides flexibility as compared to other options for where user credentials are saved, allows users to create their own accounts, and password reset functionality. Bryer (2024) <doi:10.5281/zenodo.10987876>.
License GPL (>= 3)
BugReports https://github.com/jbryer/login/issues
Imports cookies, DBI, digest, emayili, htmltools, shiny, shinybusy, shinyjs, stringr, utils
Suggests knitr, rmarkdown
Enhances RSQLite
VignetteBuilder knitr
Encoding UTF-8
RoxygenNote 7.3.1
NeedsCompilation no
Author Jason Bryer [aut, cre] (<https://orcid.org/0000-0002-2454-0402>)
Repository CRAN
Date/Publication 2024-04-18 19:32:48 UTC

R topics documented:

emayili_emailer .......................................................... 2
is_logged_in .................................................................. 3
emayili_emailer

Returns a function to send emails using the emayili package.

Description

This returns a function that can be used with the login_server(). Specifically, the function takes two parameters, to_email and message.

Usage

emayili_emailer(
    email_host = NULL,
    email_port = NULL,
    email_username = NULL,
    email_password = NULL,
    from_email = NULL
)

Arguments

email_host      SMTP email host.
email_port      SMTP email port.
email_username  username for the SMTP server.
email_password  password for the SMTP server.
from_email      the from email address sent from login_server().

Value

returns a function to send an email using the emayili package.

See Also

login_server()
isLoggedIn

Display Shiny elements only if the user is logged in.

Description
This function can be used on the Shiny UI side. It will check to see if the user is logged in, if so the other Shiny elements will be displayed.

Usage
isLoggedIn(id, ...)

Arguments
id
  id unique ID for the Shiny Login module.
...
  Shiny UI elements.

Value
  a shiny::conditionalPanel() object.

See Also
  login_server()

isNotLoggedIn

Display Shiny elements only if the user is not logged in.

Description
This function can be used on the Shiny UI side. It will check to see if the user is not logged in, if so the other Shiny elements will be displayed.

Usage
isNotLoggedIn(id, ...)

Arguments
id
  id unique ID for the Shiny Login module.
...
  Shiny UI elements.

Value
  a shiny::conditionalPanel()
See Also

login_server()

login_server

Login server module.

Description

This is the main server logic for the login Shiny module to be included in server.R side.

Usage

login_server(
  id,
  db_conn = NULL,
  users_table = "users",
  activity_table = "users_activity",
  emailer = NULL,
  new_account_subject = "Verify your new account",
  reset_password_subject = "Reset password",
  verify_email = !is.null(emailer),
  additional_fields = NULL,
  cookie_name = "loginusername",
  cookie_expiration = 30,
  username_label = "Email:",
  password_label = "Password:",
  create_account_label = "Create Account",
  create_account_message = NULL,
  reset_email_message = NULL,
  enclosing_panel = shiny::wellPanel,
  code_length = 6,
  salt = NULL,
  salt_algo = "sha512",
  shinybusy_spin = "fading-circle",
  shinybusy_position = "full-page"
)

Arguments

id unique ID for the Shiny Login module.
db_conn a DBI database connection.
users_table the name of the table in the database to store credentials.
activity_table the name of the table in the database to log login and logout activity.
emailer function used to send email messages. The function should have have three parameters: to_email for the address to send the email, subject for the subject of the email and message for the contents of the email address. See `emayilli_emailer()` for an example.

new_account_subject the subject used for verifying new accounts.

reset_password_subject the subject of password reset emails.

verify_email if true new accounts will need to verify their email address before the account is created. This is done by sending a six digit code to the email address.

additional_fields a character vector of additional fields the user is asked to fill in at the when creating a new account. The names of the vector correspond to the variable names and the values will be used as the input labels.

cookie_name the name of the cookie saved. Set to `NULL` to disable cookies.

cookie_expiration the number of days after which the cookie will expire.

username_label label used for text inputs of username.

password_label label used for text inputs of password.

create_account_label label for the create account button.

create_account_message Email message sent to confirm email when creating a new account. Include `\%s` somewhere in the message to include the code.

reset_email_message Email message sent to reset password. Include `\%s` somewhere in the message to include the code.

enclosing_panel the Shiny element that contains all the UI elements. The default is `shiny::wellPanel()`. If you wish a more subtle appearance `htmltools::div()` is a reasonable choice.

code_length the number of digits of codes emailed for creating accounts (if `verify_email == TRUE`) or resetting passwords.

salt a salt to use to encrypt the password before storing it in the database.

salt_algo the algorithm used to encrypt the password. See `digest::digest()` for more details.

shinybusy_spin Style of the spinner when sending emails. See `shinybusy::use_busy_spinner()` for more information.

shinybusy_position Position of the spinner when sending emails. See `shinybusy::use_busy_spinner()` for more information.

Value

a `shiny::reactiveValues()` object that includes two values: `logged_in` (this is `TRUE` if the user is logged in) and `username` which has the user’s login username if logged in.
Examples

```r
library(shiny)
library(login)

### User Interface
ui <- fluidPage(
  titlePanel("Shiny Login Simple Demo"),
  p("You can login with 'test/test'."),
  login::login_ui(id = 'login_demo'),
  login::logout_button('login_demo'),
  hr(),
  div('Are you logged in? ', textOutput('is_logged_in')), 
  div('Username: ', textOutput('username')), 
  login::is_logged_in( 
    id = 'login_demo',
    div("This only shows when you are logged in!")
  ),
  login::is_not_logged_in( 
    id = 'login_demo',
    div("This only shows when you are NOT logged in!")
  )
)

### Server
server <- function(input, output, session) {
  USER <- login::login_server( 
    id = 'login_demo',
    db_conn = RSQLite::dbConnect(RSQLite::SQLite(), 'users.sqlite')
  )
  observeEvent(USER$logged_in, { 
    if(USER$logged_in) {
      shinyjs::hide(id = 'login_box')
    } else {
      shinyjs::show(id = "login_box")
    }
  })
  output$is_logged_in <- renderText({
    USER$logged_in
  })
  output$username <- renderText({
    USER$username
  })
}

### Run the application
if(interactive()) {
  shinyApp(ui = ui, server = server)
}
```
login_ui  

**Login UI elements.**

**Description**

This will render (if the user is not logged in) text boxes and buttons for the user to login.

**Usage**

```r
login_ui(id)
```

**Arguments**

- `id`  
  id unique ID for the Shiny Login module.

**Value**

a `shiny::div()` object.

logout_button  

**Logout button.**

**Description**

Render a button for the user to logout.

**Usage**

```r
logout_button(id,  
  label = "Logout",  
  icon = shiny::icon("right-from-bracket"),  
  style = "",  
  check_login = TRUE
)
```

**Arguments**

- `id`  
  id unique ID for the Shiny Login module.
- `label`  
  label of the logout button.
- `icon`  
  icon for the logout button.
- `style`  
  CSS styles for the logout button.
- `check_login`  
  if TRUE this will call `is_logged_in()`.

**Value**

a `shiny::actionButton()` if the user is logged in.
new_user_ui  
*UI for creating a new user account.*

**Description**

This will render the UI for users to create an account.

**Usage**

```r
new_user_ui(id)
```

**Arguments**

- `id`  
  id unique ID for the Shiny Login module.

**Value**

shiny object containing the input fields for a user to create an account.

passwdInput  
*Password input textbox.*

**Description**

This is an extension to Shiny’s built in passwordInput by encrypting the password client side before sending it to the server. Although it is encrypted in the client using JavaScript it highly recommend that you also use an SSL certificate (for https) as well.

**Usage**

```r
passwdInput(inputId, label, value)
```

**Arguments**

- `inputId`  
  ID for the input.
- `label`  
  label for the textbox.
- `value`  
  default value.

**Value**

a `shiny::tagList()` object.
reset_password_ui

UI for resetting password.

Description
Displays UI for users to reset their password. In order for the password reset feature to work credentials to a SMTP server must be passed to the login_server() function.

Usage
reset_password_ui(id)

Arguments
id id unique ID for the Shiny Login module.

Value
a shiny object containing the input fields for a user to reset their password.

use_login
JavaScript and CSS dependencies.

Description
This ensures the JavaScript and CSS dependencies are available to the client. Files are located in assets/ folder when installed.

Usage
use_login()

Value
a htmltools::htmlDependency() object defining the JavaScript and CSS files.
Index

digest::digest(), 5
emayili_emailer, 2
emayili_emailer(), 5
htmltools::div(), 5
htmltools::htmlDependency(), 9
is_logged_in, 3
is_logged_in(), 7
is_not_logged_in, 3
login_server, 4
login_server(), 2–4, 9
login_ui, 7
logout_button, 7
new_user_ui, 8
passwdInput, 8
reset_password_ui, 9
shiny::actionButton(), 7
shiny::conditionalPanel(), 3
shiny::div(), 7
shiny::reactiveValues(), 5
shiny::tagList(), 8
shiny::wellPanel(), 5
shinybusy::use_busy.spinner(), 5
use_login, 9