Package 'shiny.reglog'

July 14, 2021

Title Optional Login and Registration Module System for ShinyApps

Version 0.2.0.2

Description Package creates UI modules for Login, Register and Password Reset boxes and server module for reading and writing user database contained within googlesheet or sqlite file. Creates automatic e-mail messages after registration and for resetting password procedure. Currently supports English language (default) and Polish. The authentication system created with shiny.reglog is designed to be optional: user don't need to be logged-in to access your application, but when logged-in the user data can be used to read from and write to relational databases.

Imports shiny, dplyr, dbplyr, scrypt, R6, DBI, googlesheets4, RSQLite, lubridate

Suggests shinydashboard, googledrive, gmailr, emayili

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1.9001

NeedsCompilation no

Author Michal Kosinski [aut, cre] (<https://orcid.org/0000-0002-8426-3654>)

Maintainer Michal Kosinski <kosinski.mich@gmail.com>

Repository CRAN

Date/Publication 2021-07-14 09:00:02 UTC

R topics documented:

create_gsheet_db 2
create_sqlite_db 2
login_server 3
login_UI 6
password_reset_UI 7
register_UI 8
use_language 9

Index **10**

create_gsheet_db *Function to create new empty 'googlesheet' database*

Description

Function to create new empty 'googlesheet' database

Usage

```
create_gsheet_db(name = NULL)
```

Arguments

name	specify name for 'googlesheet' file. Defaults to random name.

Value

id of the 'googlesheet' file. After creation you need to provide it to `login_server()`.

Examples

```
if(googlesheets4::gs4_has_token()){

gsheet.id <- create_gsheet_db()

# you can then pass 'gsheet.id' to you 'login_server' call
#
# login_server(db_method = "gsheet",
#              gsheet_file = gsheet.id,
#              ...)
#

googledrive::drive_trash(gsheet.id)

}
```

create_sqlite_db *Function to create new empty 'SQLite' database*

Description

Function to create new empty 'SQLite' database

Usage

```
create_sqlite_db(output_file)
```

Arguments

output_file	path to new 'SQLite' database. After creation you need to provide it to login_server()

Examples

```
sqlite.path <- tempfile(fileext = "sqlite")

create_sqlite_db(sqlite.path)

# you can then pass 'sqlite.path' to you 'login_server' call
#
# login_server(db_method = "sqlite",
#              sqlite_db = sqlite.path,
#              ...)
#
```

---

login_server *Login server module*

Description

Shiny server module for the optional login/registration system

This function creates a server module to handle other modules of the system: login_UI(), password_reset_UI() and register_UI

It uses database contained in 'googlesheet' file on your 'gdrive' or 'SQLite' database locally to read and write data of the users. You need to create a 'googlesheet' or 'SQLite' using create_gsheet_db() or create_sqlite_db() respectively.

Usage

```
login_server(
  id = "login_system",
  db_method,
  mail_method,
  appname,
  appaddress,
  lang = "en",
  gsheet_file,
  sqlite_db,
  gmailr_user,
  emayili_user,
  emayili_password,
  emayili_host,
  emayili_port
)
```

Arguments

id	the id of the module. Defaults to "login_system" for all of the modules contained within the package. If you plan to use serveral login systems inside your app or for any other reason need to change it, remember to keep consistent id for all elements of module.
db_method	the character string containing chosen database container, either: "gsheet" (needing installation of 'googlesheets4' package) or "sqlite" (needing installation of 'DBI' and 'RSQLite' packages)
mail_method	the character string containing chosen method of sending emails, either: "gmailr" (needing installation of 'gmailr' package) "emayili" (needing installation of 'emayili' package)
appname	the character string containing the name of your application (used in automatic e-mails for information purposes)
appaddress	the character value containing the web address of your application (used in automatic e-mails for information purposes)
lang	specifies the app used language. Accepts "en" or "pl". Defaults to "en"
gsheet_file	the ID of your 'googlesheet' file holding the database. It is contained within URL address of your googlesheet (for: db_method = "gsheet")
sqlite_db	the path to your 'SQLite' database (for: db_method = "sqlite")
gmailr_user	your gmail address (for: db_method = "gmailr")
emayili_user	your email address, also used as login to your email account (for: db_method = "emayili")
emayili_password	password to your email account (for: db_method = "emayili")
emayili_host	host of your email box (for: db_method = "emayili")
emayili_port	port of your email box (for: db_method = "emayili")

Details

The module logic creates a reactiveValues() object with loaded database of users and reset codes stored in session$userData. It allows to cut the reading from database to only one read per loading of the app - unfortunately it makes the app run slowly if the database of users gets very long.

Registration of new account mails the confirmation e-mail to the end user on provided e-mail.

Provided e-mail is needed for password reset: 10 digits code is generated and mailed to the user to confirm its identity. Reset code remains valid for 24 hours.

Authorization:

- When using db_method of "gsheet" you need to authorize access to your google drive outside of the functions (using googlesheets4:gs_auth() with default scopes: "https://www.googleapis.com/auth/spr
- When using mail_method of "emayili" you need to allow "less secure apps" to use your mailbox
- When using mail_method of "gmailr" you need to authorize access to your gmail box by creating Oauth2 App on 'Google Cloud Platform' and passing it to gmailr::gm_auth_configure() and allowing scopes: "https://www.googleapis.com/auth/gmail.send"

Security:

- Both passwords and reset codes are hashed with the help of 'scrypt' package for the extra security
- gmailr mail_method seems to be more secure if you intend to use 'gmail' account to send emails. 'emayili' is suggested only when using other mailboxes.

Value

reactiveValues() object with three elements:

`is_logged`, containing boolean describing authorization status

`user_id`, containing the logged user identification name. When not logged, it contains the timestamp of session start

`user_mail`, containing the logged user mail. When not logged, it is empty character string of nchar() value 0: ("")

See Also

`login_UI()` for the login window in UI

`password_reset_UI()` for the password reset window in UI

`register_UI()` for the registration window in UI

Examples

```
# NOT RUN {
## Only run this example in interactive R sessions

if(interactive()){

#### example of db_method = "sqlite" and mail_method = "emayili"

library(shiny)
library(shiny.reglog)

# Define UI containing shiny.reglog modules
ui <- fluidPage(

    headerPanel(title = "shiny.reglog test"),

    tabsetPanel(
        tabPanel("Values",
                 # table of returned data for active user
                 dataTableOutput("active_user_values"),
             # table of session$userData$reactive_db$user_db loaded at the start of session
                 dataTableOutput("user_db"),
            # table of session$userData$reactive_db$reset_db loaded at the start of session
                 dataTableOutput("reset_db")
        ),
        tabPanel("Login", login_UI()),
        tabPanel("Register", register_UI()),
```

```
        tabPanel("Reset Password", password_reset_UI())
    )
)

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

    # login server with specified methods for database and mailing
    # to run it you need to replace placeholders with your details and
    # cofigure it for your needs

    auth <- login_server(
        db_method = "sqlite",
        mail_method = "emayili",
        appname = "shiny.reglog test",
        appaddress = "not-on-net.com",
        sqlite_db = "test.sqlite",
        emayili_user = "your_email_address",
        emayili_password = "your_email_password",
        emayili_port = "your_email_box_port",
        emayili_host = "your_email_box_host"
    )

    # table of values returned by login_server

    output$active_user_values <- renderDataTable({
        data.frame(is_logged = auth$is_logged,
                   user_id = auth$user_id,
                   user_mail = auth$user_mail
        )
    })

    # tibbles contained within session$userData$reactive_db

    output$user_db <- renderDataTable(
        session$userData$reactive_db$user_db
    )

    output$reset_db <- renderDataTable(
        session$userData$reactive_db$reset_db
    )
}

# Run the application
shinyApp(ui = ui, server = server)

}

# }
```

`login_UI` *Shiny UI module for login box*

Description

This function creates a UI div() element containing informations and input necessary for user to log-in. As it outputs a div() element, you can put it inside container of your choosing (be it some tabItem, fluidPage, fluidRow etc.)

Usage

```
login_UI(id = "login_system", lang = "en")
```

Arguments

id	the id of the module. Defaults to "login_system" for all of the modules contained within the package. If you plan to use serveral login systems inside your app or for any other reason need to change it, remember to keep consistent id for all elements of module.
lang	specifies the app used language. Accepts "en" or "pl". Defaults to "en"

Details

It need to be used in conjuction with `login_server()` function and is suggested to be used alongside `password_reset_UI()` and `register_UI()` for full potential.

Value

NONE

See Also

login_server() for more details and example

`password_reset_UI` *Shiny UI module for password reset*

Description

This function creates a UI div() element containing informations and input necessary for user to reset password. As it outputs a div() element, you can put it inside container of your choosing (be it some tabItem, fluidPage, fluidRow etc.). It is important to mention that password reset procedure invokes modalDialog(), so it should be avoided to contain this function inside one.

Usage

```
password_reset_UI(id = "login_system", lang = "en")
```

Arguments

id	the id of the module. Defaults to "login_system" for all of the modules contained within the package. If you plan to use serveral login systems inside your app or for any other reason need to change it, remember to keep consistent id for all elements of module.
lang	specifies the app used language. Accepts "en" or "pl". Defaults to "en"

Details

It need to be used in conjuction with `login_server()` function and is suggested to be used alongside `login_UI()` and `register_UI()` for full potential.

Value

NONE

See Also

login_server() for more details and example

---

register_UI *Shiny UI module for registration box*

Description

This function creates a UI div() element containing informations and input necessary for user to register new account. As it outputs a div() element, you can put it inside container of your choosing (be it some tabItem, fluidPage, fluidRow etc.)

Usage

```
register_UI(id = "login_system", lang = "en")
```

Arguments

id	the id of the module. Defaults to "login_system" for all of the modules contained within the package. If you plan to use serveral login systems inside your app or for any other reason need to change it, remember to keep consistent id for all elements of module.
lang	specifies the app used language. Accepts "en" or "pl". Defaults to "en"

Details

It need to be used in conjuction with `login_server()` function and is suggested to be used alongside `login_UI()` and `password_reset_UI()` for full potential.

Value

NONE

See Also

login_server() for more details and example

`use_language` *Function for language creation*

Description

Function for language creation

Usage

```
use_language(lan = "en")
```

Arguments

`lan`	give language

Index

create_gsheet_db, 2
create_sqlite_db, 2

login_server, 3
login_UI, 6
login_UI(), *5*

password_reset_UI, 7
password_reset_UI(), *5*

register_UI, 8
register_UI(), *5*

use_language, 9