Package ‘polished’

March 9, 2021

Type Package
Title Authentication, User Administration, and Hosting for ‘shiny’ Apps
Version 0.3.0
Maintainer Andy Merlino <andy.merlino@tychobra.com>
Description Easily add modern authentication and user administration to your ‘shiny’ apps. Customize user sign in and registration pages to match your brand. Control who can access one or more of your ‘shiny’ apps.
License MIT + file LICENSE
BugReports https://github.com/tychobra/polished/issues
Encoding UTF-8
LazyData true
Imports automagic, cli, digest, dplyr, DT, htmltools, htr, jose, jsonlite, lubridate, purrr, R6, rlang, shiny, shinycssloaders, shinydashboard, shinyFeedback, shinyjs, shinyWidgets, stats, stringr, tibble, tidyr, tools, utils, uuid, yaml
Suggests knitr, rmarkdown
VignetteBuilder knitr
RoxygenNote 7.1.1
NeedsCompilation no
Author Andy Merlino [aut, cre], Patrick Howard [aut], Jimmy Briggs [aut]
Repository CRAN
Date/Publication 2021-03-09 10:00:02 UTC
R topics documented:

admin_button_ui ................................. 2
bundle_app .................................. 3
default_admin_ui_options .................. 4
deploy_app .................................. 4
email_input .................................. 5
firebase_dependencies ...................... 6
firebase_init .................................. 7
global_sessions_config ...................... 7
password_input ................................ 9
profile_module ................................ 10
profile_module_ui ................................ 10
providers_ui .................................. 11
remove_query_string ....................... 11
secure_server ................................ 12
secure_static ................................. 13
secure_ui ..................................... 14
send_password_reset_email_module ....... 15
send_password_reset_email_module_ui .... 15
Sessions ...................................... 16
set_config_env ................................ 18
sign_in_check_jwt ............................. 18
sign_in_js .................................... 19
sign_in_module ................................ 19
sign_in_module_2 .............................. 20
sign_in_module_2_ui .......................... 20
sign_in_module_ui ............................. 21
sign_in_ui_default ............................ 21
sign_out_from_shiny ......................... 22

Index  .......................................... 23

admin_button_ui  An html button to navigate the the "Admin Panel"

Description

The UI portion of the 'shiny' module for the button to navigate to the "Admin Panel". This is the button that, when clicked, navigates a 'polished' admin from your 'shiny' app to the 'polished' Admin Panel. If your app is set up with the default 'polished' configuration, this button appears in the bottom right of your 'shiny' app.

Usage

admin_button_ui(align = "right", vertical_align = "bottom")
bundle_app

Arguments

align The horizontal alignment of the button. Valid options are "right" (the default) or "left".
vertical_align the vertical alignment of the button. Valid options are "bottom" (the default) or "top"

Value

admin button UI

bundle_app Create a tar archive

Description

This function is called by deploy_app() to compress Shiny apps before deploying them to Polished Hosting. You probably won’t need to call this function directly.

Usage

bundle_app(app_dir = ".")

Arguments

app_dir The path to the directory containing your Shiny app. Defaults to the working directory.

Examples

## Not run:
bundle_app(
  system.file("examples/polished_example_01", package = "polished")
)
## End(Not run)
### default_admin_ui_options

**Default Options for the Admin UI**

**Description**

This function specifies the default logos that are displayed in the "Admin Panel".

**Usage**

```r
default_admin_ui_options()
```

**Value**

the default list of HTML for branding elements in the Admin Panel UI. The valid list element names are:

- **title** - Title/Logo element in top left corner of Admin Panel dashboard & browser tab title
- **sidebar_branding** - Branding (e.g. Logo) on left sidebar of Admin Panel dashboard
- **browser_tab_icon** - Icon to display in browser tab

---

### deploy_app

**Deploy a Shiny app to Polished Hosting**

**Description**

Deploy a Shiny app to Polished Hosting

**Usage**

```r
defloy_app(
    app_name,
    app_dir = ".",
    api_key = getOption("polished")$api_key,
    api_url = "https://host-api.polished.tech",
    launch_browser = TRUE,
    region = "us-east1",
    ram_gb = 2,
    r_ver = NULL
)
```
Arguments

app_name  You Shiny app's name.
app_dir   The path to the directory containing your Shiny app.
api_key   Your polished.tech API key. Defaults to `getOption("polished")$api_key`.
api_url   The Polished API url. Defaults to "https://host-api.polished.tech". You should not change from the default unless you are testing a development version of the API.
launch_browser Whether or not to open your default browser to your newly deployed app after it is successfully deployed. TRUE by default.
region    the region to deploy the app to on Google Cloud Platform. See https://cloud.google.com/compute/docs/regions-zones for all available regions on Google Cloud Platform. Currently on "us-east1" is supported, but soon, all regions will be supported.
ram_gb    the amount of memory to allocate to your Shiny app server. Valid values are 2, 4, or 8.
r_ver     Character string of R version. If kept as NULL, the default, then `deploy_app()` will detect the R version you are currently running. The R version must be a version supported by an r-ver Docker image. You can see all the r-ver Docker image versions of R here https://github.com/rocker-org/rocker-versioned2/tree/master/dockerfiles and here https://github.com/rocker-org/rocker-versioned/tree/master/r-ver.

Examples

```r
## Not run:
deploy_app(
  app_name = "polished_example_01",
  app_dir = system.file("examples/polished_example_01", package = "polished"),
  api_key = "<your polished.tech API key>"
)
## End(Not run)
```

email_input  A Shiny email input

Description

This is a replica of shiny::textInput() with the HTML input type attribute set to "email" rather than "text".
Usage

```r
email_input(
  inputId,
  label = tagList(icon("envelope"), "Email"),
  value = "",
  width = NULL,
  placeholder = NULL
)
```

Arguments

- **inputId**: The input slot that will be used to access the value.
- **label**: Display label for the control, or NULL for no label.
- **value**: Initial value.
- **width**: The width of the input, e.g. '400px'.
- **placeholder**: A character string giving the user a hint as to what can be entered into the control. Internet Explorer 8 and 9 do not support this option.

---

**firebase_dependencies**  *Load the Firebase JavaScript dependencies into the UI*

Description

Under the hood, polished uses Firebase JavaScript dependencies to handle user authentication. This function loads the required Firebase JavaScript dependencies in the UI of your Shiny app.

Usage

```r
firebase_dependencies(services = c("auth"), firebase_version = "7.15.5")
```

Arguments

- **services**: character vector of Firebase services to load into the UI. Valid strings are "auth" (default), "firestore", "functions", "messaging", and "storage".
- **firebase_version**: character string of the Firebase version. Defaults to "7.15.5".

Value

- the HTML `<script>` tags for the Firebase JavaScript dependencies

Examples

```r
firebase_dependencies()
```
**firebase_init**

*Initialize Firebase*

### Description

Executes a couple lines of JavaScript to initialize Firebase. This function should be called in your Shiny UI immediately after `firebase_dependencies`.

### Usage

```r
firebase_init(firebase_config)
```

### Arguments

- `firebase_config`  
  list of Firebase configuration

### Value

a character string of JavaScript code to initialize Firebase

### Examples

```r
## Not run:
my_config <- list(
  apiKey = "your Firebase API key",
  authDomain = "your Firebase auth domain",
  projectId = "your Firebase Project ID"
)
firebase_init(my_config)
## End(Not run)
```

---

**global_sessions_config**

*Configuration for global sessions*

### Description

This is the primary function for configuring polished. It configures your app’s instance of the Sessions class that manages your user’s polished sessions. Call this function in your `global.R` file. See https://github.com/Tychobra/polished/blob/master/inst/examples/polished_example_01/global.R for a complete example.
Usage

```r
global_sessions_config(
  app_name,
  api_key,
  firebase_config = NULL,
  admin_mode = FALSE,
  is_invite_required = TRUE,
  api_url = "https://api.polished.tech",
  sign_in_providers = "email",
  is_email_verification_required = TRUE,
  is_auth_required = TRUE,
  sentry_dsn = NULL
)
```

Arguments

- `app_name` the name of the app.
- `api_key` the API key. Either from https://polished.tech or your on premise polished API deployment.
- `firebase_config` a list containing your Firebase project configuration. This list should have the following named elements:
  - `apiKey`
  - `authDomain`
  - `projectId`
- `admin_mode` FALSE by default. Set to TRUE to enter the polished Admin Panel without needing to register and sign in. This is useful during development for inviting the first users to your app. Make sure to set `admin_mode = FALSE` before deploying your app.
- `is_invite_required` TRUE by default. Whether or not to require the user to have an invite before registering/signing in
- `api_url` the API url. Defaults to "https://api.polished.tech".
- `sign_in_providers` the sign in providers to enable. Valid values are "google", "email", "microsoft", and/or "facebook". Defaults to "email".
- `is_email_verification_required` TRUE by default. Whether or not to require the user to verify their email before accessing your Shiny app.
- `is_auth_required` TRUE by default. Whether or not to require users to be signed in to access the app. It can be useful to set this argument to FALSE if you want to allow users to do certain actions (such as viewing charts and tables) without signing in, and only require users to sign in if they want to save data to your database.
- `sentry_dsn` either NULL, the default, or your Sentry project DSN.
password_input

Examples

```r
## Not run:
# global.R

global_sessions_config(
  app_name = "<your app name>",
  api_key = "<your API key>"
)

## End(Not run)
```

password_input  A modification of shiny::passwordInput

Description

This modified version of Shiny's `passwordInput()` does not actually send the password to our Shiny server. It is just a regular password input that always keeps your user's password on the client. The password is used to sign the user in and then converted to a JWT by Firebase, all on the client, before it is sent to your Shiny server.

Usage

```r
password_input(
  input_id,
  label = tagList(icon("unlock-alt"), "Password"),
  value = "",
  style = "",
  placeholder = NULL
)
```

Arguments

- **input_id**: The input slot that will be used to access the value.
- **label**: Display label for the control, or NULL for no label.
- **value**: Initial value.
- **style**: Character string of in-line CSS to style the input.
- **placeholder**: A character string giving the user a hint as to what can be entered into the control. Internet Explorer 8 and 9 do not support this option.
**profile_module**  
*Profile Module Server*

**Description**

The server logic to accompany the `profile_module_ui`.

**Usage**

```r
profile_module(input, output, session)
```

**Arguments**

- `input`: the Shiny server input
- `output`: the Shiny server output
- `session`: the Shiny server session

---

**profile_module_ui**  
*Profile Module UI*

**Description**

Generates the UI for a user profile dropdown button to be used with the `shinydashboard` package.

**Usage**

```r
profile_module_ui(id, other_lis = NULL)
```

**Arguments**

- `id`: the Shiny module id.
- `other_lis`: additional `<li>` HTML tags to place between the email address and the Sign out button in the user profile dropdown. This is often used to add a user "My Account" page/app where the user can set their account settings.
### providers_ui

**UI for the Firebase authentication providers buttons**

**Description**

Creates the HTML UI of the "Sign in with *" buttons. These buttons are only necessary if you enable social sign in via the `sign_in_providers` argument passed to `global_sessions_config`.

**Usage**

```r
providers_ui(
  ns,
  sign_in_providers = c("google", "email"),
  title = "Sign In",
  fancy = TRUE
)
```

**Arguments**

- **ns**: the Shiny namespace function created with `shiny::NS()`.
- **sign_in_providers**: the sign in providers to enable. Valid values are "google" "email", "microsoft", and/or "facebook". Defaults to "email".
- **title**: The title to be used above the provider buttons. Set to NULL to not include.
- **fancy**: Should the buttons be large and colorful?

**Value**

the HTML UI of the "Sign in with *" buttons.

### remove_query_string

**Remove the URL query**

**Description**

Remove the entire query string from the URL. This function should only be called inside the server function of your Shiny app.

**Usage**

```r
remove_query_string(
  session = shiny::getDefaultReactiveDomain(),
  mode = "replace"
)
```
secure_server

Arguments

- **session**: the Shiny session
- **mode**: the mode to pass to shiny::updateQueryString(). Valid values are "replace" or "push".

Description

This function is used to secure your Shiny app's server function. Make sure to pass your Shiny app's server function as the first argument to secure_server() at the bottom of your Shiny app's server.R file.

Usage

```r
secure_server(
  server,
  custom_sign_in_server = NULL,
  custom_admin_server = NULL,
  allow_reconnect = FALSE,
  account_module = NULL,
  splash_module = NULL
)
```

Arguments

- **server**: A Shiny server function (e.g function(input, output, session) {})
- **custom_sign_in_server**: Either NULL, the default, or a Shiny module server containing your custom sign in server logic.
- **custom_admin_server**: Either NULL, the default, or a Shiny module server function containing your custom admin server functionality.
- **allow_reconnect**: argument to pass to the Shiny session$allowReconnect() function. Defaults to FALSE. Set to TRUE to allow reconnect with shiny-server and Rstudio Connect. Set to "force" for local testing. See https://shiny.rstudio.com/articles/reconnecting.html for more information.
- **account_module**: the server code for the user account module.
- **splash_module**: the server code for the splash page.
secure_static

secure_static
Secure a static HTML page

Description

secure_static() can be used to secure any HTML page using polished. It is often used to add polished to .Rmd htmloutput and flexdashboards.

Usage

secure_static(
    html_file_path,
    global_sessions_config_args,
    sign_out_button = shiny::actionLink("sign_out", "Sign Out", icon =
        shiny::icon("sign-out-alt"), class = "polished_sign_out_link")
)

Arguments

    html_file_path  the path the to HTML file. See the details for more info.
    global_sessions_config_args
        arguments to be passed to global_sessions_config.
    sign_out_button
        action button or link with inputId = "sign_out". Set to NULL to not include a
        sign out button.

Details

To secure a static HTML page, place the HTML page in a folder named “www” and call secure_static() from a file named app.R. The file structure should look like:

- app.R
- www/
  - index.html

See an example here: https://github.com/Tychobra/polished_example_apps/tree/master/05_flex_dashboard

Value

a Shiny app object
Secure your Shiny UI

Description

This function is used to secure your Shiny app’s UI. Make sure to pass your Shiny app’s UI as the first argument to `secure_ui()` at the bottom of your Shiny app’s `ui.R` file.

Usage

```r
secure_ui(
  ui,
  sign_in_page_ui = NULL,
  custom_admin_ui = NULL,
  custom_admin_button_ui = admin_button_ui(),
  admin_ui_options = default_admin_ui_options(),
  account_module_ui = NULL,
  splash_module_ui = NULL
)
```

Arguments

- **ui**
  - UI of the application.

- **sign_in_page_ui**
  - Either `NULL`, the default (See `sign_in_ui_default`), or the HTML, CSS, and JavaScript to use for the UI of the Sign In page.

- **custom_admin_ui**
  - Either `NULL`, the default, or a list of 2 Shiny module UI functions to add additional shinydashboard tabs to the polished Admin Panel. The list must be in the form:
    ```r
    list(
      "menu_items" = <your_custom_admin_menu_ui("custom_admin">,
      "tab_items" = <your_custom_admin_tabs_ui("custom_admin">
    )
    ```

- **custom_admin_button_ui**
  - Either `admin_button_ui()`, the default, or your custom UI to take Admins from the custom Shiny app to the polished Admin Panel.

- **admin_ui_options**
  - List of HTML elements to customize branding of the polished Admin Panel. Valid list element names are `title`, `sidebar_branding`, and `browser_tab_icon`. See `default_admin_ui_options`, the default.

- **account_module_ui**
  - the UI portion for the user’s account module.

- **splash_module_ui**
  - the UI portion for the splash page module.
send_password_reset_email_module

Value
Secured Shiny app UI

Description
This function sends a request to the https://polished.tech API to reset a user’s password.

Usage
send_password_reset_email_module(input, output, session, email)

Arguments
input the Shiny server input
output the Shiny server output
session the Shiny server session
e-mail A reactive value returning the email address to send the password reset email to.

send_password_reset_email_module_ui

de UI for a Shiny module to send a password reset email

Description
de UI for a Shiny module to send a password reset email

Usage
send_password_reset_email_module_ui(id)

Arguments
id the Shiny module id
Sessions

R6 class to track polished sessions

Description

An instance of this class handles the 'polished' user sessions for each 'shiny' app using 'polished'. The 'shiny' developer should not need to interact with this class directly.

Methods

Public methods:

- Sessions$config()
- Sessions$sign_in_social()
- Sessions$get_invite_by_email()
- Sessions$find()
- Sessions$sign_in_email()
- Sessions$register_email()
- Sessions$refresh_email_verification()
- Sessions$set_signed_in_as()
- Sessions$get_signed_in_as_user()
- Sessions$set_inactive()
- Sessions$sign_out()
- Sessions$get_admin_mode()
- Sessions$clone()

Method config(): polished Sessions configuration function

Usage:
Sessions$config(
  firebase_config = NULL,
  admin_mode = FALSE,
  is_invite_required = TRUE,
  sign_in_providers = "email",
  is_email_verification_required = TRUE,
  is_auth_required = TRUE
)

Details: This function is called via `global_sessions_config()` in `global.R` of all Shiny apps using polished.

Method sign_in_social(): verify the users Firebase JWT and store the session

Usage:
Sessions$sign_in_social(firebase_token, hashed_cookie)

Arguments:
firebase_token  the Firebase JWT. This JWT is created client side (in JavaScript) via `firebase.auth()`.
hashed_cookie – the hashed polished cookie. Used for tracking the user session. This cookie is inserted into the "polished.sessions" table if the JWT is valid.

*Returns*: NULL if sign in fails. If sign in is successful, a list containing the following:
- email
- email_verified
- is_admin
- user_uid
- hashed_cookie
- session_uid

**Method** `get_invite_by_email()`:
*Usage:*
Sessions$get_invite_by_email(email)

**Method** `find()`:
*Usage:*
Sessions$find(hashed_cookie, page)

**Method** `sign_in_email()`:
*Usage:*
Sessions$sign_in_email(email, password, hashed_cookie)

**Method** `register_email()`:
*Usage:*
Sessions$register_email(email, password, hashed_cookie)

**Method** `refresh_email_verification()`:
*Usage:*
Sessions$refresh_email_verification(session_uid, firebase_token)

**Method** `set_signed_in_as()`:
*Usage:*
Sessions$set_signed_in_as(session_uid, signed_in_as, user_uid = NULL)

**Method** `get_signed_in_as_user()`:
*Usage:*
Sessions$get_signed_in_as_user(user_uid)

**Method** `set_inactive()`:
*Usage:*
Sessions$set_inactive(session_uid, user_uid)

**Method** `sign_out()`:
*Usage:*
Sessions$sign_out(hashed_cookie, session_uid)
Method `get_admin_mode()`:

*Usage:*

```r
Sessions$get_admin_mode()
```

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

*Usage:*

```r
Sessions$clone(deep = FALSE)
```

*Arguments:*

- `deep` Whether to make a deep clone.

### Description

Determines if the app is deployed to a server or running locally, and adjusts the config environment to "production" or "default", respectively. This function is almost always called in the `global.R` file of a Shiny app immediately before the configuration in the `config.yml` is read in.

*Usage*

```r
set_config_env(override = NULL)
```

*Arguments*

- `override` Set the environment to "default" or "production" manually. **CAUTION:** Be sure you know the difference between "default" & "production" configuration environments. Using the "production" environment will affect the database of the deployed application.

### Description

This function retrieves the JWT created by the JavaScript from `sign_in_js` and signs the user in as long as the token can be verified. This function should be called in the server function of a shiny module. Make sure to call `sign_in_js` in the UI function of this module.

*Usage*

```r
sign_in_check_jwt(jwt, session = shiny::getDefaultReactiveDomain())
```

*Arguments*

- `jwt` a reactive returning a Firebase JSON web token for the signed in user.
- `session` the shiny session.
**sign_in_js**

*Sign in and register pages JavaScript dependencies*

**Description**

This function should be called at the bottom of your custom sign in and registration pages UI. It loads in all the JavaScript dependencies to handle polished sign in and registration. See the vignette for details.

**Usage**

```r
sign_in_js(ns)
```

**Arguments**

- **ns**
  
  the ns function from the Shiny module that this function is called within.

---

**sign_in_module**

*Server logic for the Sign In & Register pages*

**Description**

This server logic accompanies the `sign_in_module_ui`.

**Usage**

```r
sign_in_module(input, output, session)
```

**Arguments**

- **input**
  
  the Shiny input

- **output**
  
  the Shiny output

- **session**
  
  the Shiny session
**sign_in_module_2**  
*Server logic for the Sign In & Register pages*

**Description**

This server logic accompanies `sign_in_module_2_ui`.

**Usage**

```r
sign_in_module_2(input, output, session)
```

**Arguments**

- `input` the Shiny input
- `output` the Shiny output
- `session` the Shiny session

---

**sign_in_module_2_ui**  
*UI for the Sign In & Register pages*

**Description**

Alternate sign in UI that works regardless of whether or not invites are required. The UI displays email sign in inputs on the left, and social sign in options on the right. `sign_in_module_2` must be provided as the argument `custom_sign_in_server` in `secure_server` for proper functionality.

**Usage**

```r
sign_in_module_2_ui(id)
```

**Arguments**

- `id` the Shiny module id
sign_in_module_ui

UI for the Sign In & Register pages

Description

UI for the Sign In & Register pages when a user invite is required to Register & Sign In.

Usage

```r
sign_in_module_ui(id, register_link = "First time user? Register here!")
```

Arguments

- `id`: the Shiny module id
- `register_link`: The text that will be displayed in the link to go to the user registration page. The default is "First time user? Register here!". Set to NULL if you don’t want to use the registration page.

sign_in_ui_default

Default UI styles for the Sign In & Registration pages

Description

Default styling for the sign in & registration pages. Update the `sign_in_ui_default()` arguments with your brand and colors to quickly style the sign in & registration pages to match your brand.

Usage

```r
sign_in_ui_default(
  sign_in_module = sign_in_module_ui("sign_in"),
  color = "#5ec7dd",
  company_name = "Your Brand Here",
  logo_top = tags$div(style = "width: 300px; max-width: 100%; color: #FFF;", class = "text-center", h1("Your", style = "margin-bottom: 0; margin-top: 30px;"), h1("Brand", style = "margin-bottom: 0; margin-top: 10px;"), h1("Here", style = "margin-bottom: 15px; margin-top: 10px;")),
  logo_bottom = NULL,
  icon_href = "polish/images/polished_icon.png",
  background_image = NULL,
  terms_and_privacy_footer = NULL,
  align = "center"
)
```
Arguments

- **sign_in_module**: UI module for the Sign In & Registration pages.
- **color**: hex color for the background and button.
- **company_name**: your company name.
- **logo_top**: HTML for logo to go above the sign in panel.
- **logo_bottom**: HTML for the logo below the sign in panel.
- **icon_href**: the URL/path to the browser tab icon.
- **background_image**: the URL/path to a full width background image. If set to NULL, the default, the color argument will be used for the background instead of this image.
- **terms_and_privacy_footer**: links to place in the footer, directly above the copyright notice.
- **align**: The horizontal alignment of the Sign In box. Defaults to "center". Valid values are "left", "center", or "right".

Value

the UI for the Sign In & Registration pages

---

**sign_out_from_shiny**  
*Sign Out from your Shiny app*

Description

Call this function to sign a user out of your Shiny app. This function should be called inside the server function of your Shiny app. See [https://github.com/Tychobra/polished/blob/master/inst/examples/polished_example_01/server.R](https://github.com/Tychobra/polished/blob/master/inst/examples/polished_example_01/server.R) For an example of this function being called after the user clicks a "Sign Out" button.

Usage

```r
sign_out_from_shiny(
  session = shiny::getDefaultReactiveDomain(),
  redirect_page = "?page=sign_in"
)
```

Arguments

- **session**: the Shiny session
- **redirect_page**: the query string for the page that the user should be redirected to after signing out.
Index

admin_button_ui, 2
bundle_app, 3
default_admin_ui_options, 4, 14
deploy_app, 4
email_input, 5
firebase_dependencies, 6, 7
firebase_init, 7
global_sessions_config, 7, 11, 13
password_input, 9
profile_module, 10
profile_module_ui, 10, 10
providers_ui, 11
remove_query_string, 11
secure_server, 12, 20
secure_static, 13
secure_ui, 14
send_password_reset_email_module, 15
send_password_reset_email_module_ui, 15
Sessions, 16
set_config_env, 18
sign_in_check_jwt, 18
sign_in_js, 18, 19
sign_in_module, 19
sign_in_module_2, 20, 20
sign_in_module_2_ui, 20, 20
sign_in_module_ui, 19, 21
sign_in_ui_default, 14, 21
sign_out_from_shiny, 22