Package ‘polished’

February 24, 2022

Type Package

Title Authentication, User Administration, and Hosting for 'shiny' Apps

Version 0.6.1

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. Also, deploy & host your apps with Polished Hosting.

License MIT + file LICENSE


BugReports https://github.com/tychobra/polished/issues

Encoding UTF-8

Imports automagic, digest, dplyr, DT, htmltools, htr, jose, jsonlite, lubridate, purrr, rlang, rmarkdown, shiny, shinydashboard, shinyFeedback, shinyjs, shinyWidgets, stats, stringr, tibble, tidyr, utils, uuid, yaml

Suggests testthat (>= 3.0.0), knitr, config, xfun

VignetteBuilder knitr

RoxygenNote 7.1.2

Config/testthat/edition 3

NeedsCompilation no

Author Andy Merlino [aut, cre],
        Patrick Howard [aut],
        Jimmy Briggs [aut]

Repository CRAN

Date/Publication 2022-02-24 11:50:06 UTC
R topics documented:

add_app .......................................................... 3
add_app_user ..................................................... 3
add_role ........................................................... 4
add_user ........................................................... 5
add_user_role ..................................................... 5
admin_button_ui .................................................. 6
api_list_to_df ................................................... 6
bundle_app ......................................................... 7
default_admin_ui_options ...................................... 7
delete_app ........................................................ 8
delete_app_user ................................................... 9
delete_role ........................................................ 9
delete_user ........................................................ 10
delete_user_role .................................................. 10
deploy_app ......................................................... 11
e-mail_input ....................................................... 12
firebase_dependencies .......................................... 13
firebase_init ..................................................... 13
get_apps .......................................................... 14
get_app_users .................................................... 15
get_dependent_packages ....................................... 16
get_roles .......................................................... 17
get_users .......................................................... 17
get_user_roles ................................................... 18
password_input ................................................... 19
polished_api_res ................................................ 20
polished_config .................................................. 20
print.polished_api_res .......................................... 22
profile_module ................................................... 23
profile_module_ui ............................................... 23
providers_ui ...................................................... 24
remove_query_string ............................................ 24
secure_rmd ....................................................... 25
secure_server .................................................... 26
secure_static ..................................................... 27
secure_ui .......................................................... 28
send_password_reset_email_module ................................ 29
send_password_reset_email_module_ui ................................ 29
set_api_key ....................................................... 30
set_config_env ................................................... 30
sign_in_check_jwt ............................................... 31
sign_in_js ........................................................ 31
sign_in_module ................................................... 32
sign_in_module_2 ............................................... 32
sign_in_module_2_ui ............................................ 33
sign_in_module_ui ............................................... 33
add_app

Description
Polished API - Add an App

Usage
add_app(app_name, app_url = NULL, api_key = get_api_key())

Arguments
app_name the app name.
app_url an optional app url. This url will be included in links sent out in invite and email verification emails to redirect your users to your app.
api_key your Polished API key. Set your polished api key using set_api_key() so that you do not need to supply this argument with each function call.

See Also
get_apps() update_app() delete_app()

add_app_user

Description
Polished API - Add a User to an App

Usage
add_app_user(
  app_uid,
  user_uid = NULL,
  email = NULL,
  is_admin = FALSE,
  send_invite_email = FALSE,
  api_key = get_api_key()
)
Arguments

- **app_uid**: the app uid.
- **user_uid**: an optional user uid for the user to be invited to the app.
- **email**: an optional email address for the user to be invited to the app.
- **is_admin**: boolean (default: FALSE) - whether or not the user is a Polished admin.
- **send_invite_email**: boolean - whether or not to send the user an invite email notifying them they have been invited to access the app.
- **api_key**: your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.

Details

Supply either the `user_uid` or `email`. If both are provided, then the `user_uid` will be used, and the `email` will be ignored.

See Also

- `get_app_users()`
- `update_app_user()`
- `delete_app_user()`

---

**add_role**

Polished API - Add a Role

Description

Polished API - Add a Role

Usage

```
add_role(role_name, api_key = get_api_key())
```

Arguments

- **role_name**: a role name.
- **api_key**: your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.

See Also

- `get_roles()`
- `delete_role()`
### add_user

**Polished API - Add a User**

**Description**
Polished API - Add a User

**Usage**

```
add_user(email, api_key = get_api_key())
```

**Arguments**

- `email`: the new user’s email address.
- `api_key`: your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.

**See Also**

- `get_users()`
- `delete_user()`

### add_user_role

**Polished API - Add a User Role**

**Description**
Polished API - Add a User Role

**Usage**

```
add_user_role(user_uid, role_uid, api_key = get_api_key())
```

**Arguments**

- `user_uid`: a user uid.
- `role_uid`: a role name.
- `api_key`: your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.

**See Also**

- `get_user_roles()`
- `delete_user_role()`
### 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**

```r
admin_button_ui(align = "right", vertical_align = "bottom")
```

**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

### api_list_to_df

Convert a list returned from the Polished API into a data frame

**Description**

In order to avoid issues with converting R data frames into JSON objects and back to R data frames, we instead convert R data frames to R lists before converting them to JSON to be sent via the Polished API. This function then converts those lists back into R data frames (or more precisely tibbles).

**Usage**

```r
api_list_to_df(api_list)
```

**Arguments**

- **api_list**
  
  a list. All elements in the list are vectors of the same length.

**Value**

a tibble
**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**

```r
bundle_app(app_dir = ".")
```

**Arguments**

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

**Examples**

```r
## 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

Polished API - Delete an App

delete_app

Description

Polished API - Delete an App

Usage

delete_app(app_uid = NULL, app_name = NULL, api_key = get_api_key())

Arguments

app_uid an optional app uid. One of either app_uid or app_name must be provided.
app_name an optional app name. One of either app_uid or app_name must be provided.
api_key your Polished API key. Set your polished api key using set_api_key() so that you do not need to supply this argument with each function call.

Details

If both app_uid and app_name arguments are provided, then the app_uid will be used and the app_name will be ignored.

See Also

get_apps() add_app() update_app()
**Polished API - Delete an App User**

**Description**
Polished API - Delete an App User

**Usage**
```
delete_app_user(app_uid, user_uid, api_key = get_api_key())
```

**Arguments**
- **app_uid**: an app uid.
- **user_uid**: a user uid.
- **api_key**: your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.

**See Also**
```
get_apps() add_app() update_app_user()
```

**Polished API - Delete a Role**

**Description**
Polished API - Delete a Role

**Usage**
```
delete_role(role_uid, api_key = get_api_key())
```

**Arguments**
- **role_uid**: the role uid of the role to be deleted.
- **api_key**: your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.

**See Also**
```
get_roles() add_role()
```
**delete_user**  
*Polished API - Delete a User*

**Description**
Polished API - Delete a User

**Usage**
```
delete_user(user_uid, api_key = get_api_key())
```

**Arguments**
- **user_uid**: the uid of the user to be deleted.
- **api_key**: your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.

**See Also**
`get_users()`, `add_user()`

---

**delete_user_role**  
*Polished API - Delete a User Role*

**Description**
Polished API - Delete a User Role

**Usage**
```
delete_user_role(role_uid, user_uid, api_key = get_api_key())
```

**Arguments**
- **role_uid**: the role uid of the role to be deleted.
- **user_uid**: the user uid that the role should be removed from.
- **api_key**: your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.

**See Also**
`get_user_roles()`, `add_user_role()`
deploy_app

**Description**

Deploy a Shiny app to Polished Hosting

**Usage**

```r
deploy_app(
  app_name,
  app_dir = ".",
  api_key = get_api_key(),
  launch_browser = TRUE,
  region = "us-east1",
  ram_gb = 2,
  r_ver = NULL,
  tlmgr = character(0),
  golem_package_name = NULL,
  cache = TRUE
)
```

**Arguments**

- **app_name**: Your Shiny app’s name.
- **app_dir**: The path to the directory containing your Shiny app.
- **api_key**: Your polished API key. Defaults to `Sys.getenv("POLISHED_API_KEY")` if set.
- **launch_browser**: Boolean (default: TRUE) - Whether or not to open your newly deployed app in your default web browser after successful deployment.
- **region**: the region to deploy the app to on Google Cloud Platform. See [https://cloud.google.com/run/docs/locations](https://cloud.google.com/run/docs/locations) for all available regions on Google Cloud Platform. Currently, database connections are only supported for us-east1. See [https://polished.tech/docs/06-database-connections](https://polished.tech/docs/06-database-connections) for details.
- **ram_gb**: the amount of memory (in GiB) to allocate to your Shiny app’s server. Valid values are 2, 4, or 8.
- **r_ver**: Character string of desired R version. If kept as NULL (the default), `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](https://github.com/rocker-org/rocker-versioned2/tree/master/dockerfiles) and here [https://github.com/rocker-org/rocker-versioned/tree/master/r-ver](https://github.com/rocker-org/rocker-versioned/tree/master/r-ver).
- **tlmgr**: a character vector of TeX Live packages to install. This is only used if your Shiny app generates PDF documents. Defaults to `character(0)` for no TeX Live installation. Provide a character vector of your TeX Live package dependencies to have all your TeX Live packages installed at build time.
golem_package_name

if your Shiny app was created as a package with the golem package, provide the name of the Shiny app package as a character string. Defaults to NULL. Keep as NULL for non golem Shiny apps.

cache

Boolean (default: TRUE) - whether or not to cache the Docker image.

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
eemail_input(
    inputId,
    label = tagList(shiny::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 Social sign in & user authentication when `sign_in_providers` besides "email" are included in `polished_config()`. This function loads the required Firebase JavaScript dependencies in the UI of your Shiny app.

**Usage**
```
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**
```
firebase_dependencies()
```

**firebase_init**  *Initialize Firebase*

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

**Usage**
```
firebase_init(firebase_config)
```
get_apps

Arguments

firebase_config

named list of firebase configuration values. Required values are:

  • apiKey
  • authDomain
  • projectId

Value

a character string of JavaScript code to initialize Firebase

Examples

## 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)

get_apps

Polished API - Get App(s)

Description

Polished API - Get App(s)

Usage

get_apps(app_uid = NULL, app_name = NULL, api_key = get_api_key())

Arguments

app_uid an optional app uid.
app_name an optional app name.
api_key your Polished API key. Set your polished api key using set_api_key() so that you do not need to supply this argument with each function call.
Details

If both the app_uid and app_name are NULL, then all the apps in your account will be returned. If either app_uid or app_name are not NULL, then a single app will be returned (assuming the app exists). If both the app_uid and app_name are provided, then the app_uid will be used, and the app_name will be ignored. If the app does not exists, a zero row tibble will be returned.

Value

an object of class polished_api_res. The content of the object is a tibble of app(s) with the following columns:

- uid
- app_name
- app_url
- created_at
- modified_at

See Also

add_app() update_app() delete_app()
get_dependent_packages

Details

If app_uid, user_uid, & email are all NULL, then all app users will be returned.

Value

an object of class polished_api_res. The content of the object is a tibble of app(s) with the following columns:

• uid
• app_uid
• user_uid
• is_admin
• created_at
• email

See Also

add_app_user() update_app_user() delete_app_user()

description

Note: this function is copied from the automagic R package. We are including it in polished while we await the merging of this PR https://github.com/cole-brokamp/automagic/pull/17 and a new CRAN release of automagic.

Usage

get_dependent_packages(directory = getwd())

Arguments

directory folder to search for R and Rmd files

Details

parses all R and Rmd files in a directory and uses automagic::parse_packages to find all R packages required for the code to run

Value

a vector of package names
get_roles

**Polished API - Get Role(s)**

**Description**
Polished API - Get Role(s)

**Usage**
```r
get_roles(role_uid = NULL, api_key = get_api_key())
```

**Arguments**
- **role_uid**: an optional role uid.
- **api_key**: your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.

**Value**
an object of class `polished_api_res`. The content of the object is a tibble of user(s) with the following columns:
- uid
- role_name
- created_at

**See Also**
- `add_role()`
- `delete_role()`

get_users

**Polished API - Get User(s)**

**Description**
Polished API - Get User(s)

**Usage**
```r
get_users(user_uid = NULL, email = NULL, api_key = get_api_key())
```

**Arguments**
- **user_uid**: an optional user uid.
- **email**: an optional user email.
- **api_key**: your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.
Details

If both the user_uid and email are NULL, then all the users in your account will be returned. If either user_uid or email are not NULL, then a single user will be returned (assuming the user exists). If both the user_uid and email are provided, then the user_uid will be used, and the email will be ignored. If the user does not exists, a zero row tibble will be returned.

Value

an object of class polished_api_res. The content of the object is a tibble of users(s) with the following columns:

• uid
• email
• email_verified
• created_by
• created_at
• modified_by
• modified_at
• is_password_set

See Also

add_user() delete_user()
**Value**

an object of class polished_api_res. The "content" of the object is a tibble of users(s) with the following columns:

- role_uid
- role_name
- user_uid
- user_name
- created_at

**See Also**

`add_user_role()` `delete_user_role()`

**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 = htmltools::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.
polished_api_res  

Send GET Request to the Polished API

**Description**
Send GET Request to the Polished API

**Usage**
```r
polished_api_res(resp)
```

**Arguments**
- `resp`  
a Polished API response

**Value**
- an S3 object of class "polished_api_res".

---

polished_config  

*global configuration for polished authentication*

**Description**
*global configuration for polished authentication*

**Usage**
```r
polished_config(
  app_name,
  api_key = get_api_key(),
  firebase_config = NULL,
  admin_mode = FALSE,
  is_invite_required = TRUE,
  sign_in_providers = "email",
  is_email_verification_required = TRUE,
  sentry_dsn = NULL,
  cookieExpires = 365L,
  is_auth_required = TRUE
)

global_sessions_config(...)
```
Arguments

app_name
the name of the Shiny app.

api_key
the polished API key, available at https://dashboard.polished.tech.

firebase_config
if using Social Sign In (see https://polished.tech/docs/03-social-sign-in for more documentation), a list containing your Firebase project configuration (Default: NULL). 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

sign_in_providers
a character vector of 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.

sentry_dsn
either NULL, the default, or your Sentry project’s DSN.

cookieExpires
the number of days before a user’s cookie expires. Set to NULL to force Sign Out at session end. This argument is passed to the expires option in js-cookie: https://github.com/js-cookie/js-cookie#expires. Default value is 365L (i.e. 1 year)

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.

... arguments to pass to polished_config

Details

This is the primary function for configuring polished. It configures your app’s instance of the Polished class that manages polished authentication. 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.

Examples
## Not run:
# global.R

polished_config(
  app_name = "<your app name>",
  api_key = "<your API key>",
  firebase_config = list(
    apiKey = "<Firebase apiKey>",
    authDomain = "<Firebase authDomain",
    projectId = "<Firebase projectId>"
  ),
  sign_in_providers = c(
    "email",
    "google",
    "microsoft"
  )
)

## End(Not run)

---

**print.polished_api_res**

**print polished_api_res**

---

**Description**

Generic print function for polished_api_res S3 class.

**Usage**

```r
# S3 method for class 'polished_api_res'
print(x, ...)
```

**Arguments**

- `x` an S3 object of class `polished_api_res`.
- `...` additional arguments.
profile_module  Profile Module Server

Description

The server logic to accompany the profile_module_ui.

Usage

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

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 Social Sign In 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 `polished_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**  
  a character vector of 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"
)
```
Arguments

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

secure_rmd Render and secure R Markdown document

Description

secure_rmd() can be used to render (or run) and secure many types of R Markdown documents. Rendering is handled either by rmarkdown::render or, if using shiny, a shiny app is constructed, and then the output is secured with polished authentication.

Usage

secure_rmd(
  rmd_file_path,
  polished_config_args = list(),
  sign_in_page_args = list(),
  sign_out_button = NULL
)

Arguments

rmd_file_path the path to .Rmd file.
polished_config_args arguments to be passed to polished_config. (NOTE: Values passed in this list will override YAML header values if both provided).

sign_in_page_args a named list() to customize the Sign In page UI. Valid names are color, company_name, logo, & background_image. (NOTE: Values passed in this list will override YAML header values if both provided).

sign_out_button A shiny::actionButton or shiny::actionLink with inputId = "sign_out". If this argument is left as NULL, secure_rmd will attempt to add in an appropriate sign out button/link depending on the output format of your .Rmd document. Set this argument to list() to not include a sign out button.

Value

a Shiny app object
Examples

```r
## Not run:

secure_rmd(system.file("examples/rmds/flexdashboard.Rmd", package = "polished"))
secure_rmd(
  system.file("examples/rmds/flexdashboard.Rmd", package = "polished"),
  polished_config_args = list(
    # any values in this list will override values in YAML header
    app_name = "different_name"
  ),
  sign_in_page_args = list(
    color = "#FF5700"
  )
)
secure_rmd(system.file("examples/rmds/flexdashboard_shiny.Rmd", package = "polished"))
secure_rmd(system.file("examples/rmds/html_document.Rmd", package = "polished"))
secure_rmd(system.file("examples/rmds/pdf_document.Rmd", package = "polished"))
io_file_path <- system.file("examples/rmds/ioslides/ioslides_presentation.Rmd", package = "polished"
)
secure_rmd(io_file_path)

## End(Not run)
```

---

`secure_server`  
Secure your Shiny app’s server

**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,
  override_user = TRUE
)
```

**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.

override_user
whether or not to override the session$user with the polished session$userData$user user. By default this is now set to TRUE, but if you are using a hosting option that uses the session$user (e.g. RStudio Connect), then you may want to set this to FALSE. The polished user can always be found at session$userData$user().

---

### 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

```r
secure_static(
    html_file_path,
    polished_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.
- **polished_config_args** arguments to be passed to `polished_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

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
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()
)

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:

list(
  "menu_items" = <your_custom_admin_menu_ui("custom_admin">,
  "tab_items" = <your_custom_admin_tabs_ui("custom_admin">
)
send_password_reset_email_module

The server logic for a Shiny module to send a password reset email

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This function sends a request to the <a href="https://polished.tech">https://polished.tech</a> API to reset a user’s password.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>send_password_reset_email_module(input, output, session, email)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>input: the Shiny server input</td>
</tr>
<tr>
<td>output: the Shiny server output</td>
</tr>
<tr>
<td>session: the Shiny server session</td>
</tr>
<tr>
<td>email: A reactive value returning the email address to send the password reset email to.</td>
</tr>
</tbody>
</table>

send_password_reset_email_module_ui

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

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>the UI for a Shiny module to send a password reset email</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>send_password_reset_email_module_ui(id)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>id: the Shiny module id</td>
</tr>
</tbody>
</table>
set_api_key  set Polished API key

Description
The API key can be set as an Environment Variable via Sys.getenv("POLISHED_API_KEY").

Usage

```r
set_api_key(api_key)
get_api_key()
```

Arguments

- `api_key`  the Polished API key

Value

- a list of the newly set polished R options

Examples

```r
set_api_key(api_key = "<my Polished API key>")
```

set_config_env  Automatically set the config environment

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.
sign_in_check_jwt

Check the JWT from the user sign in

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**

```
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**

```
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_social**

verify the users Firebase JWT and store the session

**Description**

verify the users Firebase JWT and store the session

**Usage**

```python
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.

**Value**

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

**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

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",
    button_color = NULL
)

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"

button_color  the color of the "Continue", "Sign In", and "Register" buttons. If kept as NULL, the default, then the button color will be the same color as the color passed to the color argument.

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 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.

update_app  Polished API - Update an App

Description

Polished API - Update an App

Usage

```r
update_app(app_uid, app_name = NULL, app_url = NULL, api_key = get_api_key())
```

Arguments

- `app_uid`: the app uid of the app to update.
- `app_name`: an optional app name to replace the existing app name.
- `app_url`: an optional app url to replace the existing app url.
- `api_key`: your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.

See Also

- `get_apps()`  `add_app()`  `delete_app()`
Polished API - Update an App User

**Description**

Polished API - Update an App User

**Usage**

```r
update_app_user(app_uid, user_uid, is_admin = FALSE, api_key = get_api_key())
```

**Arguments**

- **app_uid** the app uid to update.
- **user_uid** the user uid to update.
- **is_admin** boolean (default: FALSE) - whether or not the user is an admin.
- **api_key** your Polished API key. Set your polished api key using `set_api_key()` so that you do not need to supply this argument with each function call.

**See Also**

- `get_app_users()`
- `add_app_user()`
- `delete_app_user()`
Index

add_app, 3
add_app(), 8, 9, 15, 36
add_app_user, 3
add_app_user(), 16, 37
add_role, 4
add_role(), 9, 17
add_user, 10, 18
add_user_role, 5
add_user_role(), 10, 19
admin_button_ui, 6
api_list_to_df, 6
bundle_app, 7
default_admin_ui_options, 7, 29
delete_app, 8
delete_app(), 3, 15, 36
delete_app_user, 9
delete_app_user(), 4, 16, 37
delete_role, 9
delete_role(), 4, 17
delete_user, 10
delete_user(), 5, 18
delete_user_role, 10
delete_user_role(), 5, 19
deploy_app, 11
eemail_input, 12
firebase_dependencies, 13, 13
firebase_init, 13
get_api_key(set_api_key), 30
gget_app_users, 15
gget_app_users(), 4, 37
gget_apps, 14
gget_apps(), 3, 8, 9, 36
gget_dependent_packages, 16
gget_roles, 17
gget_roles(), 4, 9
gget_user_roles, 18
gget_user_roles(), 5, 10
gget_users, 17
gget_users(), 5, 10
gglobal_sessions_config
(set_polished_config), 20
gpassword_input, 19
gpolished_api_res, 20
gpolished_config, 20, 21, 24, 25, 27
gprint.polished_api_res, 22
gprofile_module, 23
gprofile_module_ui, 23, 23
getProviders_ui, 24
geremove_query_string, 24
get_secure_rmd, 25
gsecure_server, 26, 33
get_secure_static, 27
get_secure_ui, 28
gsend_password_reset_email_module, 29
gsend_password_reset_email_module_ui, 29
gset_api_key, 3–5, 8–10, 14, 15, 17, 18, 30, 36, 37
gset_config_env, 30
gsign_in_check_jwt, 31
gsign_in_js, 31, 31
gsign_in_module, 32
gsign_in_module_2, 32, 33
gsign_in_module_2_ui, 32, 33
gsign_in_module_ui, 32, 33
gsign_in_social, 34
gsign_in_ui_default, 28, 34
gsign_out_from_shiny, 36
gupdate_app, 36
gupdate_app(), 3, 8, 15
gupdate_app_user, 37
gupdate_app_user(), 4, 9, 16