

Package ‘brochure’

October 14, 2026

Title Multipage 'Shiny' Apps

Version 1.0.0

Description Build natively multi-page 'Shiny' applications, serving content on several endpoints instead of a single page. Pages are declared with their own user interface and server function and are dispatched by URL with 'routr', including parameterised routes. Redirections, request and response middleware, and helpers to read and set cookies are also provided.

License MIT + file LICENSE

URL <https://github.com/colinfay/brochure>,
<https://github.com/ColinFay/brochure>

BugReports <https://github.com/ColinFay/brochure/issues>

Imports attempt (>= 0.3.1), htmltools, purrr, rlang (>= 0.4.12), routr (>= 2.0.0), shiny (>= 1.6.0)

Suggests golem (>= 0.3.0), knitr, rmarkdown, testthat (>= 3.0.0), usethis, withr

Config/testthat/edition 3

VignetteBuilder knitr

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Colin Fay [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-7343-1846>)

Maintainer Colin Fay <contact@colinfay.me>

Repository CRAN

Date/Publication 2026-10-14 11:10:22 UTC

Contents

brochureApp	2
get_keys	4
golem_hook	5
new_page	6
page	6
parse_cookie_string	8
redirect	9
server_redirect	10
set_cookie	11
Index	13

brochureApp	<i>Create a brochureApp</i>
-------------	-----------------------------

Description

This function is to be used in place of shinyApp(). It takes a series of page()s, each with its own url, UI and server function, and serves them from a single Shiny application.

Usage

```
brochureApp(  
  ...,  
  onStart = NULL,  
  options = list(),  
  enableBookmarking = NULL,  
  content_404 = "Not found",  
  basepath = "",  
  req_handlers = list(),  
  res_handlers = list(),  
  wrapped = shiny::tagList  
)
```

Arguments

...	The page()s and redirect()s of the app, plus anything to inject as is into every page: tags, tag lists, html dependencies and strings. That is how you add something to all your pages at once, a <script> or the resources of a {golem} app for example. Note that a string is injected too, so a stray value ends up rendered on every page. Anything else is an error naming the element it cannot use. A bare list is spliced, so brochureApp(list(page_1(), page_2())) builds two pages rather than injecting the list into each of them.
onStart	A function that will be called before the app is actually run. This is only needed for shinyAppObj, since in the shinyAppDir case, a global.R file can be used for this purpose.

options	Named options that should be passed to the runApp call (these can be any of the following: "port", "launch.browser", "host", "quiet", "display.mode" and "test.mode"). You can also specify width and height parameters which provide a hint to the embedding environment about the ideal height/width for the app.
enableBookmarking	Can be one of "url", "server", or "disable". The default value, NULL, will respect the setting from any previous calls to enableBookmarking() . See enableBookmarking() for more information on bookmarking your app.
content_404	The content served when no page() matches the url. A string, or anything as.character() renders as html, a tagList() for example. It is sent as it is written: unlike a page it is not rewritten for basepath, so a link in it has to carry the mount itself.
basepath	The path your app is served under by a reverse proxy. It is removed from the incoming url, so that what is left matches the href of your page(), and it is prepended to the urls the app emits. For example, if your app is served at https://connect.thinkr.fr/brochure/ and your page is named page1, use basepath = "brochure".
req_handlers	a list of functions that can manipulate the req object. These functions should take req as a parameters, and return the req object (potentially modified), or an object of class httpResponse. If any of the req_handlers return an httpResponse, this response will be sent to the browser immediately, stopping any other code.
res_handlers	A list of functions that can manipulate the httpResponse object before it is send to the browser. Each function must take a res and req parameter.
wrapped	A function taking the UI of a page and returning the UI actually served. This is how every page gets the same shell: wrapped = fluidPage gives them all the Bootstrap layout and its dependencies, and a function of your own can add a navbar or a footer around each of them. Default is shiny::tagList, which adds nothing.

Details

Behind a reverse proxy, basepath tells the app where it is mounted: it is removed from the incoming path, and prepended to the urls the app emits. On Posit Connect the mount is picked up on its own, from the RStudio-Connect-App-Base-URL header, and basepath is not needed. That header is not part of any published contract, so set basepath if you want the behaviour pinned. Brochure also injects a small script registering the handler server_redirect() talks to, which prefixes internal targets with the mount.

Value

A shiny.appobj

See Also

[page\(\)](#) to declare a page, [redirect\(\)](#) to answer an url with a redirection, and [vignette\("deployment"\)](#) to serve the app under a prefix.

Examples

```
library(shiny)

app <- brochureApp(
  page(
    href = "/",
    ui = tagList(h1("Home")),
    server = function(input, output, session) {}
  ),
  page(
    href = "/contact",
    ui = tagList(h1("Contact"))
  ),
  redirect(from = "/index.html", to = "/")
)

# Then run it as you would any Shiny app:
if (interactive()) {
  shiny::runApp(app)
}
```

get_keys*Get the keys of a parameterised page*

Description

A page declared with a parameterised href (e.g. `"/who/:id"`) exposes the values matched in the URL through `get_keys()`. Call it with no argument inside a page server, and pass it the request inside a page ui.

Usage

```
get_keys(x = shiny::getDefaultReactiveDomain())
```

Arguments

x A shiny session, or the request object a page ui receives. Defaults to the session the code is running in.

Value

A named list of the values matched in the page href, or NULL outside of a brochure page.

Examples

```
library(shiny)
page(
  href = "/who/:id",
  ui = function(request) {
```

```

      h1(get_keys(request)$id)
    },
    server = function(input, output, session) {
      print(get_keys())
    }
  )

```

golem_hook

*Golem Hook function***Description**

Golem Hook function

Usage

```
golem_hook(path, package_name, ...)
```

Arguments

path	Name of the folder to create the package in. This will also be used as the package name.
package_name	Package name to use. By default, {golem} uses basename(path). If path == '.' & package_name is not explicitly set, then basename(getwd()) will be used.
...	Arguments passed from create_golem(), unused in the default function.

Value

Used for side effect

Examples

```

if (requireNamespace("golem") & interactive()) {
  golem::create_golem(file.path(tempdir(), "myapp"), project_hook = golem_hook)
}

```

new_page

Add page

Description

Module template for golem

Usage

```
new_page(name, path, export, ...)
```

Arguments

name	The name of the module.
path	The path to the R script where the module will be written. Note that this path will not be set by the user but via <code>add_module()</code> .
export	Should the module be exported? Default is FALSE.
...	Arguments to be passed to the <code>module_template</code> function.

Value

Used for side effect

Examples

```
if (requireNamespace("golem") & interactive()) {
  golem::add_module(name = "home", module_template = brochure::new_page)
}
```

page

A Brochure Page

Description

A page is an url, a UI and a server function. Opening it starts a Shiny session of its own, separate from any other page of the app.

Usage

```
page(
  href,
  ui = tagList(),
  server = function(input, output, session) {
  },
  method = "GET",
  req_handlers = list(),
  res_handlers = list()
)
```

Arguments

href	The endpoint to serve the UI on. It can carry parameters, as in <code>"/who/:id"</code> , which are read back with <code>get_keys()</code> . See details.
ui	The UI definition of the app (for example, a call to <code>fluidPage()</code> with nested controls). If bookmarking is enabled (see <code>enableBookmarking</code>), this must be a single argument function that returns the UI definition.
server	A function with three parameters: <code>input</code> , <code>output</code> , and <code>session</code> . The function is called once for each session ensuring that each app is independent.
method	The HTTP method the page answers to. Defaults to <code>"GET"</code> . A page on another method is unreachable if the app has a <code>www/</code> directory at its root: Shiny mounts it as a static path on <code>/</code> , and <code>httpuv</code> answers anything that is not a GET or a HEAD from there with a 400 before R sees the request. Serve those files under a prefix instead, with <code>shiny::addResourcePath()</code> .
req_handlers	a list of functions that can manipulate the req object. These functions should take req as a parameters, and return the req object (potentially modified), or an object of class <code>httpResponse</code> . If any of the <code>req_handlers</code> return an <code>httpResponse</code> , this response will be sent to the browser immediately, stopping any other code.
res_handlers	A list of functions that can manipulate the <code>httpResponse</code> object before it is send to the browser. Each function must take a <code>res</code> and <code>req</code> parameter.

Details

Requests are matched against hrefs by the **routr** package, so an href is written the way `routr` writes a path:

- `"/contact"` matches that path and nothing else.
- `":name"` matches exactly one segment and captures it, so `"/who/:id"` matches `/who/colin` but neither `/who` nor `/who/colin/edit`. Use several of them if you need to: `"/pair/:a/:b"`.
- `"*"` matches whatever is left, so `"/files/*"` matches `/files/a/b/c`. The captured value is named `*1`.

A trailing slash never matters: `/contact` and `/contact/` are the same page.

Pages are tried in the order you passed them to `brochureApp()`, and the first match wins. That holds for the session a page opens as well, which is resolved on the path alone: the websocket handshake is a GET whatever method the page answers on, so two pages sharing an href run the server of the first one declared, whichever of them was served. That matters when two hrefs can match the same url: declared as `page("/who/me")` then `page("/who/:id")`, a request for `/who/me` gets the first; declared the other way round, the parameterised page catches it and the static one is never reached. Put the specific ones first.

See the **routr** documentation at <https://routr.data-imaginist.com/> for the full path syntax.

Value

A `brochure_page` object, to be passed to `brochureApp()`.

See Also

[get_keys\(\)](#) to read the parameters of an href, and [vignette\("handlers"\)](#) for req_handlers and res_handlers.

Examples

```
library(shiny)
page(
  href = "/page2",
  ui = tagList(
    h1("This is my second page"),
    plotOutput("plotb")
  )
)
```

parse_cookie_string	<i>Read the cookies sent with a request</i>
---------------------	---

Description

[get_cookies\(\)](#) returns the raw Cookie header of the request the session was opened with, and [parse_cookie_string\(\)](#) turns that header into a named vector.

Usage

```
parse_cookie_string(cookie_string)

get_cookies(session = shiny::getDefaultReactiveDomain())
```

Arguments

cookie_string	The cookie string to parse, as get_cookies() returns it.
session	The {shiny} session object.

Value

For [get_cookies\(\)](#), the Cookie header as a single string, or NULL when the request carried none. For [parse_cookie_string\(\)](#), a named character vector of the cookies it holds, or "" when given NULL. Index it with single brackets: a visitor arriving without a cookie gives nothing to index into, and [[] errors where [] returns NA.

See Also

[set_cookie\(\)](#) to set one from a response handler, and [vignette\("cookies"\)](#) for carrying a session across pages.

Examples

```
parse_cookie_string("brochure_session=63422; brochure_cookie=3958")
```

`redirect`*Redirection*

Description

Answers an url with an HTTP redirection, before any Shiny code runs. To redirect from inside a page server instead, see [server_redirect\(\)](#).

Usage

```
redirect(from, to, code = 301, method = "GET")
```

Arguments

<code>from</code>	the url to redirect from
<code>to</code>	the url to redirect to
<code>code</code>	redirecting http code, one of 301, 302, 303, 307 or 308
<code>method</code>	The HTTP method the redirection answers to. Defaults to "GET".

Value

A redirect object, to be passed to [brochureApp\(\)](#).

See Also

[server_redirect\(\)](#) to redirect from inside a page server.

Examples

```
redirect(  
  from = "/index.html",  
  to = "/"  
)  
  
# 301 says the move is permanent, 302 that it is temporary  
redirect(  
  from = "/old",  
  to = "/new",  
  code = 302  
)
```

server_redirect	<i>Do a server side redirection</i>
-----------------	-------------------------------------

Description

Do a server side redirection

Usage

```
server_redirect(to, session = shiny::getDefaultReactiveDomain())
```

Arguments

to	the destination of the redirection: a path ("/page2") or an http(s) URL. Other schemes are rejected.
session	shiny session object, default is shiny::getDefaultReactiveDomain()

Value

Used for side effect

See Also

[redirect\(\)](#) to answer an url with an HTTP redirection instead.

Examples

```
library(shiny)

# `server_redirect()` is called from a page server, so it needs a session:
page(
  href = "/login",
  ui = tagList(actionButton("go", "Take me home")),
  server = function(input, output, session) {
    observeEvent(input$go, {
      server_redirect("/")
    })
  }
)
```

`set_cookie`*Middleware to set cookies*

Description

Please read <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Set-Cookie> for more information. Description of parameters is taken from this page.

Usage

```
set_cookie(  
    res,  
    name,  
    value,  
    expires = NULL,  
    max_age = NULL,  
    domain = NULL,  
    path = NULL,  
    secure = NULL,  
    http_only = TRUE,  
    same_site = "Lax"  
)
```

```
remove_cookie(res, name, path = NULL, domain = NULL)
```

Arguments

<code>res</code>	An <code>httpResponse</code> object
<code>name</code>	A cookie-name can be any US-ASCII characters, except control characters, spaces, or tabs. It also must not contain a separator character like the following: () < > @ , ; : \ " / [] ? = { }. []: R:%20
<code>value</code>	A cookie-value can optionally be wrapped in double quotes and include any US-ASCII characters excluding control characters, Whitespace, double quotes, comma, semicolon, and backslash.
<code>expires</code>	The maximum lifetime of the cookie as an HTTP-date timestamp. Please enter an ISO 8601 datetime format.
<code>max_age</code>	Number of seconds until the cookie expires. A zero or negative number will expire the cookie immediately. If both Expires and Max-Age are set, Max-Age has precedence.
<code>domain</code>	Host to which the cookie will be sent.
<code>path</code>	A path that must exist in the requested URL, or the browser won't send the Cookie header. <code>remove_cookie()</code> only deletes a cookie when it is given the same path and domain that <code>set_cookie()</code> was given.

secure	Cookie is only sent to the server when a request is made with the https: scheme (except on localhost), and therefore is more resistant to man-in-the-middle attacks.
http_only	Forbids JavaScript from accessing the cookie, for example, through the Document.cookie property. Defaults to TRUE.
same_site	Controls whether a cookie is sent with cross-origin requests, providing some protection against cross-site request forgery attacks (CSRF). Defaults to "Lax".

Value

the httpResponse, with a cookie header

See Also

[get_cookies\(\)](#) to read them back, and [vignette\("cookies"\)](#) for carrying a session across pages.

Examples

```
set_cookie(  
  shiny::httpResponse(),  
  "this",  
  12  
)
```

Index

brochureApp, [2](#)
brochureApp(), [7](#), [9](#)

enableBookmarking(), [3](#)

get_cookies(parse_cookie_string), [8](#)
get_cookies(), [12](#)
get_keys, [4](#)
get_keys(), [7](#), [8](#)
golem_hook, [5](#)

new_page, [6](#)

page, [6](#)
page(), [3](#)
parse_cookie_string, [8](#)

redirect, [9](#)
redirect(), [3](#), [10](#)
remove_cookie(set_cookie), [11](#)

server_redirect, [10](#)
server_redirect(), [9](#)
set_cookie, [11](#)
set_cookie(), [8](#)
shiny::addResourcePath(), [7](#)