

# Package ‘shiny.router’

October 14, 2022

**Type** Package

**Title** Basic Routing for Shiny Web Applications

**Version** 0.2.3

**Description** It is a simple router for your Shiny apps.

The router allows you to create dynamic web applications with real-time User Interface and easily share url to pages within your Shiny apps.

**Encoding** UTF-8

**License** MIT + file LICENSE

**Imports** shiny, htmltools

**RoxygenNote** 7.1.1

**Suggests** testthat, covr

**NeedsCompilation** no

**Author** Filip Stachura [aut],  
Dominik Krzemiński [aut],  
Krystian Igras [aut],  
Developers Appsilon [cre, aut],  
Servet Ahmet Çizmeli [ctb],  
Appsilon [cph]

**Maintainer** Developers Appsilon <support+opensource@appsilon.com>

**Repository** CRAN

**Date/Publication** 2022-07-01 16:30:01 UTC

## R topics documented:

attach_attribs . . . . .	2
callback_mapping . . . . .	2
change_page . . . . .	3
cleanup_hashpath . . . . .	3
create_router_callback . . . . .	4
disable_bootstrap_on_bookmark . . . . .	4
extract_link_name . . . . .	5

get_page . . . . .	5
get_query_param . . . . .	6
get_url_hash . . . . .	6
is_page . . . . .	7
log_msg . . . . .	7
make_router . . . . .	8
page404 . . . . .	8
PAGE_404_ROUTE . . . . .	9
parse_url_path . . . . .	9
route . . . . .	10
router_server . . . . .	11
router_ui . . . . .	11
route_link . . . . .	12
valid_path . . . . .	12
%o:::%o . . . . .	13

## Index 14

---

attach_attribs	<i>Attach 'router-hidden' class to single page UI content</i>
----------------	---

---

### Description

Covered UI types are Shiny/htmltools tags or tag lists and html templates. In case of tag list (`tagList`) and html template (`htmlTemplate`) 'div' wrapper with 'router-hidden' class is added.

### Usage

```
attach_attribs(ui, path)
```

### Arguments

ui	Single page UI content created with proper html tags or tag list.
path	Single page path name. Attached to data-path attribute.

---

callback_mapping	<i>Create a mapping between a UI element and a server callback.</i>
------------------	---

---

### Description

Create a mapping between a UI element and a server callback.

### Usage

```
callback_mapping(path, ui, server = NA)
```

**Arguments**

path	Bookmark id.
ui	Valid Shiny user interface.
server	Function that is called within the global server function if given

**Value**

list with ui and server fields

---

change_page	<i>Change the currently displayed page.</i>
-------------	---

---

**Description**

Works by sending a message up to our reactive input binding on the clientside, which tells page.js to update the window URL accordingly, then tells clientside shiny that our reactive input binding has changed, then that comes back down to our router callback function and all other observers watching get\_page() or similar.

**Usage**

```
change_page(page, session = shiny::getDefaultReactiveDomain(), mode = "push")
```

**Arguments**

page	The new URL to go to. Should just be the path component of the URL, with optional query, e.g. "/learner?id=%d"
session	The current Shiny session.
mode	("replace" or "push") whether to replace current history or push a new one. More in shiny::updateQueryString.

---

cleanup_hashpath	<i>Formats a URL fragment into a hashpath starting with "#!/"</i>
------------------	---

---

**Description**

Formats a URL fragment into a hashpath starting with "#!/"

**Usage**

```
cleanup_hashpath(hashpath)
```

**Arguments**

hashpath	character with hash path
----------	--------------------------

**Value**

character with formatted hashpath

---

create\_router\_callback

*Internal function creating a router callback function. One need to call router callback with Shiny input and output in server code.*

---

**Description**

Internal function creating a router callback function. One need to call router callback with Shiny input and output in server code.

**Usage**

```
create_router_callback(root, routes)
```

**Arguments**

root	Main route to which all invalid routes should redirect.
routes	A routes (list).

**Value**

Router callback.

---

disable\_bootstrap\_on\_bookmark

*Fix conflicts when some bookmark uses bootstrap*

---

**Description**

This function dynamically removes bootstrap dependency when user opens specified bookmark. It should be inserted in head of bootstrap page.

**Usage**

```
disable_bootstrap_on_bookmark(bookmark)
```

**Arguments**

bookmark	Bookmark name on which bootstrap dependency should be suppressed.
----------	---

---

extract_link_name	<i>Extract link name</i>
-------------------	--------------------------

---

**Description**

Strips off the first 3 character, assuming that they are: "#!/".

**Usage**

```
extract_link_name(path)
```

**Arguments**

path	character with link path
------	--------------------------

**Value**

stripped link

---

get_page	<i>Convenience function to retrieve just the "page" part of the input.</i>
----------	--

---

**Description**

This corresponds to what might be called the "path" component of a URL, except that we're using URLs with hashes before the path & query (e.g.: <http://www.example.com/#!/virtual/path?and=params>)

**Usage**

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

**Arguments**

session	The current Shiny Session
---------	---------------------------

**Value**

The current page in a length-1 character vector, or FALSE if the input has no value.

---

get_query_param	<i>Get Query Parameters</i>
-----------------	-----------------------------

---

**Description**

Convenience function to retrieve any params that were part of the requested page. The param values returned come from "httr::parse\_url()"

**Usage**

```
get_query_param(field = NULL, session = shiny::getDefaultReactiveDomain())
```

**Arguments**

field	If provided, retrieve only a param with this name. (Otherwise, return all params)
session	The Shiny session

**Value**

The full list of params on the URL (if any), as a list. Or, the single requested param (if present). Or NULL if there's no input, or no params.

---

get_url_hash	<i>Internal function to get url hash with #!</i>
--------------	--

---

**Description**

Internal function to get url hash with #!.

**Usage**

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

**Arguments**

session	The current Shiny Session
---------	---------------------------

**Value**

Reactive hash value.

---

is_page	<i>Is page</i>
---------	----------------

---

### Description

Tell the reactive chain to halt if we're not on the specified page. Useful for making sure we don't waste cycles re-rendering the UI for pages that are not currently displayed.

### Usage

```
is_page(page, session = shiny::getDefaultReactiveDomain(), ...)
```

### Arguments

page	The page to display. Should match one of the paths sent to the
session	Shiny session
...	Other parameters are sent through to shiny::req() router.

---

log_msg	<i>Helper function to print out log messages into Shiny using cat() and stderr(), as described on <a href="https://shiny.rstudio.com/articles/debugging.html">https://shiny.rstudio.com/articles/debugging.html</a></i>
---------	---

---

### Description

Because this can print a lot, it's silent unless the shiny.router.debug option is set.

### Usage

```
log_msg(...)
```

### Arguments

...	All params get passed through to cat(). They're automatically wrapped in shiny::isolate(), so you can print reactive values here without too much worry.
-----	--

---

make_router	<i>Creates router.</i>
-------------	------------------------

---

**Description**

Returned callback needs to be called within Shiny server code.

**Usage**

```
make_router(default, ..., page_404 = page404())
```

**Arguments**

default	Main route to which all invalid routes should redirect.
...	All other routes defined with shiny.router::route function.
page_404	Styling of page when wrong bookmark is open. See <a href="#">page404</a> .

**Value**

Shiny router callback that should be run in server code with Shiny input and output lists.

**Examples**

```
## Not run:
router <- make_router(
  route("/", root_page),
  route("/other", other_page),
  page_404 = page404(
    message404 = "Please check if you passed correct bookmark name!")
)

## End(Not run)
```

---

page404	<i>404 page</i>
---------	-----------------

---

**Description**

The page which appear when path is wrong.

**Usage**

```
page404(page = NULL, message404 = NULL)
```



**Arguments**

page shiny page style, eg. `shiny::tags$div(h1("Not found"))`  
 message404 message to display at the 404 website

**Examples**

```
page404() # shiny::tags$div(h1("Not found"))
page404(message404 = "ABC") # shiny::tags$div(h1("ABC"))
```

---

PAGE_404_ROUTE	<i>Default 404 page</i>
----------------	-------------------------

---

**Description**

This is default 404 page.

**Usage**

```
PAGE_404_ROUTE
```

**Format**

An object of class character of length 1.

---

parse_url_path	<i>Parse url and build GET parameters list</i>
----------------	--

---

**Description**

Extract info about url path and parameters that follow ? sign.

**Usage**

```
parse_url_path(url_path)
```

**Arguments**

url\_path character with link url

**Details**

parse\_url\_path allows parsing paramaters lists from url. See more in examples.

Note that having query string appear before #! may cause browser to refresh and thus reset Shiny session.

**Value**

list containing two objects:

- path
- query, a list

**Examples**

```
parse_url_path("?a=1&b=foo")
parse_url_path("?a=1&b[1]=foo&b[2]=bar/#!/")
parse_url_path("?a=1&b[1]=foo&b[2]=bar/#!/other_page")
parse_url_path("www.foo.bar/#!/other_page")
parse_url_path("www.foo.bar?a=1&b[1]=foo&b[2]=bar/#!/other")
parse_url_path("#!/a=1&b[1]=foo&b[2]=bar")
parse_url_path("#!/other_page?a=1&b[1]=foo&b[2]=bar")
parse_url_path("www.foo.bar/#!/other?a=1&b[1]=foo&b[2]=bar")
```

---

route

*Create single route configuration.*

---

**Description**

Create single route configuration.

**Usage**

```
route(path, ui, server = NA)
```

**Arguments**

path	Website route.
ui	Valid Shiny user interface.
server	Function that is called as callback on server side

**Value**

A route configuration.

**Examples**

```
## Not run:
route("/", shiny::tags$div(shiny::tags$span("Hello world")))

route("/main", shiny::tags$div(h1("Main page"), p("Lorem ipsum.")))

## End(Not run)
```

---

router_server	<i>Create router pages server callback</i>
---------------	--

---

**Description**

Create router pages server callback

**Usage**

```
router_server(router)
```

**Arguments**

router Router pages object. See [make\\_router](#).

---

router_ui	<i>Creates router UI</i>
-----------	--------------------------

---

**Description**

Creates router UI

**Usage**

```
router_ui(router)
```

**Arguments**

router Router pages object. See [make\\_router](#).

**Value**

list with shiny tags that adds "router-page-wrapper" div and embeds router javascript script.

---

route_link	<i>Route link</i>
------------	-------------------

---

**Description**

Adds `/#!/` prefix to link.

**Usage**

```
route_link(path)
```

**Arguments**

path	character with path
------	---------------------

**Value**

route link

**Examples**

```
route_link("abc") # /#!/abc
```

---

valid_path	<i>Internal function that validates that path is defined in routes.</i>
------------	---

---

**Description**

Internal function that validates that path is defined in routes.

**Usage**

```
valid_path(routes, path)
```

**Arguments**

routes	A routes (list).
path	A path.

**Value**

Boolean value indicating if path is defined.



# Index

## \* datasets

PAGE\_404\_ROUTE, 9  
%::%, 13

attach\_attribs, 2

callback\_mapping, 2  
change\_page, 3  
cleanup\_hashpath, 3  
create\_router\_callback, 4

disable\_bootstrap\_on\_bookmark, 4

extract\_link\_name, 5

get\_page, 5  
get\_query\_param, 6  
get\_url\_hash, 6

is\_page, 7

log\_msg, 7

make\_router, 8, 11

page404, 8, 8  
PAGE\_404\_ROUTE, 9  
parse\_url\_path, 9

route, 10  
route\_link, 12  
router\_server, 11  
router\_ui, 11

valid\_path, 12