Package 'shiny.router'

November 19, 2020

Type Package

Title Basic Routing for Shiny Web Applications

Version 0.2.0

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

LazyData true

License MIT + file LICENSE

Imports magrittr, shiny, htmltools

RoxygenNote 7.1.1

Suggests testthat, covr

NeedsCompilation no

Author Filip Stachura [aut],
Dominik Krzemiński [cre, aut],
Krystian Igras [aut],
Appsilon [cph]

Maintainer Dominik Krzemiński <dominik@appsilon.com>

Repository CRAN

Date/Publication 2020-11-19 13:30:02 UTC

R topics documented:

`attach_attribs` *Attach 'router-hidden' class to single page UI content*

Description

Covered UI types are Shiny/htmltools tags or tag lists and httml 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` attriubute.

`callback_mapping` *Create a mapping between a UI element and a server callback.*

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 *Change the currently displayed page.*

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

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 bootrstrap page.

Usage

```
disable_bootstrap_on_bookmark(bookmark)
```

Arguments

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

extract_link_name *Extract link name*

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 *Convenience function to retrieve just the "page" part of the input.*

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 *Get Query Parameters*

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 *Internal function to get url hash with #!.*

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` *Is page*

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` *Helper function to print out log messages into Shiny using cat() and stderr(), as described on https://shiny.rstudio.com/articles/debugging.html*

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 *Creates router.*

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 page404.

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 *404 page*

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 *Default 404 page*

Description

This is default 404 page.

Usage

```
PAGE_404_ROUTE
```

Format

An object of class character of length 1.

---

parse_url_path *Parse url and build GET parameters list*

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 *Create router pages server callback*

Description

Create router pages server callback

Usage

```
router_server(router)
```

Arguments

router	Router pages object. See make_router.

router_ui *Creates router UI*

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 *Route link*

Description

Adds /#!/ prefix to link.

Usage

```
route_link(path)
```

Arguments

path	character with path

Value

route link

Examples

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

valid_path *Internal function that validates that path is defined in routes.*

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.

`%:::%` *::: hack solution*

Description

::: hack solution

Usage

```
pkg %:::% name
```

Arguments

`pkg`	package name
`name`	function name

Value

function

Index