Package 'shiny.router'

April 18, 2023

Type Package

Title Basic Routing for Shiny Web Applications

Version 0.3.1

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.

URL https://appsilon.github.io/shiny.router/,
https://github.com/Appsilon/shiny.router

BugReports https://github.com/Appsilon/shiny.router/issues

Encoding UTF-8

License MIT + file LICENSE

Imports htmltools, glue, rlang, shiny

RoxygenNote 7.2.3

Suggests covr, lintr, rcmdcheck, spelling, testthat

NeedsCompilation no

Author Ryszard Szymański [cre, aut],
Jakub Nowicki [aut],
Filip Stachura [aut],
Dominik Krzemiński [aut],
Krystian Igras [aut],
Servet Ahmet Çizmeli [ctb],
Appsilon Sp. z o.o. [cph]

Maintainer Ryszard Szymański <opensource+ryszard@appsilon.com>

Repository CRAN

Date/Publication 2023-04-18 08:00:02 UTC

R topics documented:

`change_page` *Change the currently displayed page.*

Description

Works by sending a message up to our reactive input binding on the client side, which tells page.js to update the window URL accordingly, then tells client side 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`.

`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.

---

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.

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.

make_router *[Deprecated] 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, e.g. 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 parameters 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 [deprecated]

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

Server part of the router.

Usage

```
router_server(root_page = "/", env = parent.frame())
```

Arguments

root_page	Main page path.
env	Environment (only for advanced usage).

Value

Router pages server callback.

Examples

```
## Not run:
  server <- function(input, output, session) {
    router_server(root_page = "/")
  }

## End(Not run)
```

`router_ui` *Create router UI*

Description

Creates router UI in Shiny applications.

Usage

```
router_ui(default, ..., page_404 = page404(), env = parent.frame())
```

Arguments

`default`	Main route to which all invalid routes should redirect.
`...`	All other routes defined with shiny.router::route function. It's possible to pass routes in dynamic way with dynamic dots. See `dynamic-dots` and example below
`page_404`	Styling of page when invalid route is open. See page404.
`env`	Environment (only for advanced usage), makes it possible to use shiny.router inside shiny modules.

Details

If you are defining the router inside a shiny module, we assume that the namespacing function defined in the UI is named as `ns`.

Value

Application UI wrapped in a router.

Examples

```
## Not run:
  ui <- function() {
    router_ui(
      route("/", root_page(id = "root")),
      route("other", other_page(id = "other")),
      page_404 = page404(
        message404 = "Please check if you passed correct bookmark name!")
    )
  }

## End(Not run)
## Not run:
  # create the list of routes
  dynamic_routes <- list(
    route("other2", other_page(id = "other2")),
    route("other3", other_page(id = "other3"))
  )

  ui <- function() {
    router_ui(
      route("/", root_page(id = "root")),
      route("other", other_page(id = "other")),
    # then it's possible to inject a list of arguments into a function call using rlang::`!!!`
      !!!dynamic_routes,
      page_404 = page404(
        message404 = "Please check if you passed correct bookmark name!")
    )
  }

## End(Not run)
```

`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
```

Index