Package ‘shiny.router’

September 18, 2018

Type  Package
Title  Basic Routing for Shiny Web Applications
Version  0.1.1
Author  Filip Stachura <filip@appsilon.com>
Maintainer  Dominik Krzemiński <dominik@appsilon.com>
Description  The minimal router for your Shiny apps.
It allows you to create dynamic web applications with real-time user interface and easily share URLs to pages within your Shiny apps.
Encoding  UTF-8
LazyData  true
License  MIT + file LICENSE
Imports  magrittr, shiny
RoxygenNote  6.0.1
Suggests  testthat
NeedsCompilation  no
Repository  CRAN
Date/Publication  2018-09-18 08:50:07 UTC

R topics documented:

  callback_mapping .................................................. 2
  change_page .......................................................... 2
  cleanup_hashpath ............................................... 3
  create_router_callback ........................................... 3
  extract_link_name ............................................... 4
  get_page ............................................................. 4
  get_query_param .................................................. 5
  is_page .............................................................. 5
  log_msg ............................................................. 6
  make_router ....................................................... 6
  page404 ............................................................. 7
callback_mapping  

Create a mapping between a ui element and a server callack

Description

Create a mapping between a ui element and a server callack

Usage

callback_mapping(ui, server = NA)

Arguments

ui  
Valid Shiny user interface.

server  
Function that is called within the global server function if given

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

```r
cleanup_hashpath(hashpath)
```

#### Arguments

- **hashpath**: character with hash path

### 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

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

#### Arguments

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

#### Value

Router callback.
### extract_link_name

**Extract link name**

**Description**

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

**Usage**

```r
evaluate(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. 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)**

**Description**

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

```r
evaluate(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

**Description**

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

**Usage**

```r
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

**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**

```r
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. Returned callback needs to be called within Shiny server code.

Description

Creates router. Returned callback needs to be called within Shiny server code.

Usage

make_router(default, ...)

Arguments

default Main route to which all invalid routes should redirect.
... All other routes defined with shiny.router::route function.

Value

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

Examples

```r
# Not run:
router <- make_router(
  route("/", root_page),
  route("/other", other_page)
)
```

```
# End(Not run)
```
Description

The page which appear when path is wrong.

Usage

page404(page = NULL, message404 = NULL)

Arguments

page | shiny page style, eg. `div(h1("Not found"))`

message404 | message to display at the 404 website

Examples

page404() # div(h1("Not found"))
page404(message404 = "ABC") # div(h1("ABC"))

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**

```python
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.

**Value**

- list containing two objects:
  - path
  - query, a list

**Examples**

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

---

route  
*Create single route configuration.*

**Description**

Create single route configuration.

**Usage**

```python
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**

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

router("/main", div(h1("Main page"), p("Lorem ipsum.")))

## End(Not run)
```

---

**router_ui**

*Creates an output for router. This configures client side. Call it in your UI Shiny code. In this output ui is going to be rendered according to current routing.*

**Description**

Creates an output for router. This configures client side. Call it in your UI Shiny code. In this output ui is going to be rendered according to current routing.

**Usage**

```r
router_ui()
```

**Value**

Shiny tags that configure router and build reactive but hidden input _location_.

**Examples**

```r
## Not run:
ui <- shinyUI(fluidPage(  
  router_ui()  
))

## End(Not run)
```
route_link

*Route link Adds /#!/ prefix to link.*

**Description**
Route link Adds /#!/ prefix to link.

**Usage**
```python
route_link(path)
```

**Arguments**
- `path` character with path

**Value**
route link

**Examples**
```python
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**
```python
valid_path(routes, path)
```

**Arguments**
- `routes` A routes (list).
- `path` A path.

**Value**
Boolean value indicating if path is defined.
Index

*Topic datasets
   PAGE_404_ROUTE, 7

callback_mapping, 2
change_page, 2
cleanup_hashpath, 3
create_router_callback, 3

extract_link_name, 4
get_page, 4
get_query_param, 5

is_page, 5

log_msg, 6

make_router, 6

page404, 7
PAGE_404_ROUTE, 7
parse_url_path, 8

route, 8
route_link, 10
router_ui, 9

valid_path, 10