Package ‘scenes’

January 13, 2023

Title Switch Between Alternative ‘shiny’ UIs
Version 0.1.0
Description Sometimes it is useful to serve up alternative ‘shiny’ UIs depending on information passed in the request object, such as the value of a cookie or a query parameter. This package facilitates such switches.
License MIT + file LICENSE
Encoding UTF-8
RoxygenNote 7.2.3
BugReports https://github.com/r4ds/scenes/issues
Imports cli, cookies, glue, purrr, rlang, shiny
Suggests covr, knitr, pkgload, rmarkdown, stringr, testthat (>= 3.0.0)
Config/testthat/edition 3
VignetteBuilder knitr
NeedsCompilation no
Author Jon Harmon [aut, cre, cph] (<https://orcid.org/0000-0003-4781-4346>)
Maintainer Jon Harmon <jonthegeek@gmail.com>
Repository CRAN
Date/Publication 2023-01-13 09:30:02 UTC

R topics documented:

change_scene .......................... 2
construct_action .......................... 2
default_ui .................................. 3
req_has_cookie .................................. 4
req_has_query .................................. 5
req_uses_method .................................. 6
set_scene .................................. 6

Index 8
change_scene  

**Description**

Specify a function that uses actions and the request object to choose which Shiny UI to server.

**Usage**

```
change_scene(..., fall_through = default_ui())
```

**Arguments**

- `...`  
  One or more `shiny_scene` objects.
- `fall_through`  
  A ui to display if no scenes are valid. The default value, `default_ui()`, returns an HTTP 422 status code indicating that the request cannot be processed.

**Value**

A function that processes the request object to deliver a Shiny ui.

**Examples**

```r
scene1 <- set_scene(
  "A shiny ui",
  req_has_query("scene", 1)
)
scene2 <- set_scene(
  "Another shiny ui",
  req_has_query("scene", 2)
)

ui <- change_scene(
  scene1,
  scene2
)
```

**construct_action**  

**Description**

Generate the check function for an action, and use it to create a scene_action object.
Usage

construct_action(fn, ..., negate = FALSE, methods = "GET")

Arguments

fn A function that takes a request (and potentially other arguments) and returns
TRUE or FALSE.
...
Additional parameters passed on to fn.
negate If TRUE, trigger the corresponding scene when this action is not matched.
methods The http methods which needs to be accepted in order for this function to make
sense. Default "GET" should work in almost all cases.

Value

A scene_action object.

Examples

simple_function <- function(request) {
  !missing(request) && length(request) > 0
}
sample_action <- construct_action(simple_function)
sample_action$check_fn()
sample_action$check_fn(list())
sample_action$check_fn(list(a = 1))
req_has_cookie  

Switch Scenes on Cookies

Description
Create a scene_action specifying a cookie that must be present (or absent) and optionally a check function for that cookie.

Usage
req_has_cookie(cookie_name, validation_fn = NULL, ..., negate = FALSE)

Arguments
- `cookie_name`: The cookie that must be present, as a length-1 character vector.
- `validation_fn`: A function that takes the value of the cookie as the first parameter, and returns TRUE if the cookie is valid, and FALSE otherwise.
- `...`: Additional parameters passed on to validation_fn.
- `negate`: If TRUE, trigger the corresponding scene when this action is not matched.

Value
A scene_action object, to be used in set_scene().

Examples

# Specify an action to detect a cookie named "mycookie".
req_has_cookie("mycookie")

# Specify an action to detect the *lack* of a cookie named "mycookie".
req_has_cookie("mycookie", negate = TRUE)

# Specify an action to detect a cookie named "mycookie" that has 27 characters.
req_has_cookie(
  cookie_name = "mycookie",
  validation_fn = function(cookie_value) {nchar(cookie_value == 27)}
)

# Specify an action to detect a cookie named "mycookie" that has N characters. This would make more sense in a case where validation_fn isn't an anonymous function.
req_has_cookie(
  cookie_name = "mycookie",
  validation_fn = function(cookie_value, N) {nchar(cookie_value) == N}
Description

Create a scene_action specifying a key that must be present (or absent) in the query string (the part of the URL when the shiny app was called, after "?")), and optionally a value or values for that key. For example, in myapps.shinyapps.io/myapp?param1=1&param2=text, ?param1=1&param2=text is the query string, param1 and param2 are keys, and 1 and text are their corresponding values.

Usage

req_has_query(key, values = NULL, negate = FALSE)

Arguments

key The key that must be present, as a length-1 character vector.
values Details about what to look for in the key. NULL indicates that the key must be present but its contents are unimportant for this action. Otherwise the actual value of the query must be present in values.
negate If TRUE, trigger the corresponding scene when this action is not matched.

Value

A scene_action object, to be used in set_scene().

Examples

# Specify an action to detect a "code" parameter in the query.
req_has_query("code")

# Specify an action to detect the *lack* of a "code" parameter in the query.
req_has_query("code", negate = TRUE)

# Specify an action to detect a "language" parameter, with values containing 
# "en" or "es".
req_has_query("language", "en|es")
req_uses_method  Switch Scenes on Method

Description
Create a scene_action specifying the HTTP method that must be used (or not used).

Usage
req_uses_method(method, negate = FALSE)
req_uses_get(negate = FALSE)
req_uses_post(negate = FALSE)

Arguments
method  The expected HTTP method.
negate  If TRUE, trigger the corresponding scene when this action is not matched.

Value
A scene_action object, to be used in set_scene().

Examples
req_uses_method("GET")
req_uses_method("POST")
req_uses_get()
req_uses_get(negate = TRUE)
req_uses_post()
req_uses_post(negate = TRUE)

set_scene  Link a UI to Required Actions

Description
A scene is a shiny ui and the actions that trigger it.

Usage
set_scene(ui, ...)


set_scene

Arguments

ui A shiny ui.
...

One or more scene_action objects.

Value

A shiny_scene object, which is a list with components ui and actions.

Examples

scene1 <- set_scene(
  "A shiny ui",
  req_has_query("scene", 1)
)
scene1
scene2 <- set_scene(
  "Another shiny ui",
  req_has_query("scene", 2)
)
scene2
Index

change_scene, 2
construct_action, 2

default_ui, 3
default_ui(), 2

req_has_cookie, 4
req_has_query, 5
req_uses_get (req_uses_method), 6
req_uses_method, 6
req_uses_post (req_uses_method), 6

set_scene, 6
set_scene(), 4–6