Package: scenes (via r-universe)

September 2, 2024

Title Switch Between Alternative 'shiny' UIs

Version 0.1.0.9000

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 packages facilitates such switches.

License MIT + file LICENSE

URL https://scenes.shinyworks.org/scenes/, https://github.com/shinyworks/scenes

BugReports https://github.com/shinyworks/scenes/issues

Imports cli, cookies, glue, purrr, rlang, shiny

Suggests covr, knitr, pkgload, rmarkdown, stringr, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.0

Repository https://shinyworks.r-universe.dev

RemoteUrl https://github.com/shinyworks/scenes

RemoteRef HEAD

RemoteSha 8c56263c34b631b0df520f277c28d45d0c27c5c0

Contents

---

change_scene *Choose Between Scenes*

Description

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

Usage

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

Arguments

`...`	One or more `shiny_scenes`.
`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

```
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
)
ui
```

construct_action *Construct a Scene 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`.

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))
```

default_ui *Default UI for Unprocessable Requests*

Description

A plain text UI that returns an HTTP status of 422, indicating that the request was well-formed, but semantically incorrect.

Usage

```
default_ui()
```

Value

A plain text UI with status code 422.

Examples

```
default_ui()
```

`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`, 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 a
# variable-defined number of characters.
expect_n_chars <- function(x, N) {
  nchar(x) == N
}
my_N <- 27 # Perhaps set by an environment variable.
req_has_cookie(
  cookie_name = "mycookie",
  validation_fn = expect_n_chars,
  N = my_N
)
```

`req_has_query` *Switch Scenes on Query*

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`, 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, 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)
```

scene_action-class scene_action *class*

Description

A scene_action object is a list with components check_fn and methods. It is used to test whether a request should trigger a particlar scene.

See Also

construct_action()

set_scene *Link a UI to Required Actions*

Description

Define a shiny_scene by linking a UI to zero or more scene_action requirements.

Usage

```
set_scene(ui, ...)
```

Arguments

ui	A shiny ui.
...	Zero or more scene_actions.

Value

A shiny_scene.

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
```

shiny_scene-class *shiny_scene class*

Description

A shiny_scene object is a list with components ui and actions. It is used to define what should display in a Shiny app in different scenarios.

See Also

set_scene()

Index