Package ‘cookies’

October 2, 2023

Title  Use Browser Cookies with 'shiny'

Version  0.2.3

Description  Browser cookies are name-value pairs that are saved in a user's browser by a website. Cookies allow websites to persist information about the user and their use of the website. Here we provide tools for working with cookies in 'shiny' apps, in part by wrapping the 'js-cookie' JavaScript library <https://github.com/js-cookie/js-cookie>.

License  MIT + file LICENSE


BugReports  https://github.com/r4ds/cookies/issues

Imports  cli, clock, glue, htmltools, httpuv, jsonlite, purrr, rlang, shiny (>= 1.6.0), stats, vctrs

Suggests  covr, pkgdown, rmarkdown, roxygen2, testthat (>= 3.0.0)

Config/testthat/edition  3

Encoding  UTF-8

RoxygenNote  7.2.3

NeedsCompilation  no

Author  Jon Harmon [aut, cre, cph] (<https://orcid.org/0000-0003-4781-4346>), Klaus Hartl [cph] (js-cookie JavaScript library), Fagner Brack [cph] (js-cookie JavaScript library)

Maintainer  Jon Harmon <jonthegeek@gmail.com>

Repository  CRAN

Date/Publication  2023-10-02 17:20:02 UTC

R topics documented:

  add_cookie_handlers ......................................................... 2
  cookie_dependency .......................................................... 2
  extract_cookie ............................................................... 3
add_cookie_handlers

Description

Wrap a shiny ui in this function in order to add cookie-handling functionality. The ui can be defined in any format compatible with shiny, using functions such as `shiny::fluidPage()`, `shiny::bootstrapPage()`, `shiny::htmlTemplate()`, or a raw HTML string.

Usage

```r
add_cookie_handlers(ui)
```

Arguments

- `ui` A 0- or 1-argument function defining the ui of a shiny app, or a `shiny::tagList()`.

Value

An object with the same signature as the input ui, but with the dependencies needed to handle cookies. If `ui` is a `shiny::tagList()`, a `shiny::tagList()` will be returned; if `ui` is a function, a function will be returned.

Examples

```r
str(add_cookie_handlers("example"))
```

cookie_dependency

Description

Add the js-cookie Javascript library as an HTML dependency, and make cookies available in the shiny input object.

Usage

```r
cookie_dependency()
```
extract_cookie

Details
Call this function within your shiny ui to attach the necessary JavaScript code.

Value
An htmltools::htmlDependency(), which shiny uses to add the js-cookie Javascript library exactly once.

Examples

cookie_dependency()

extract_cookie

Extract an individual cookie from a shiny request

Description
The shiny request object includes any cookies that are available to the app. This function extracts the value of a named cookie from that request.

Usage
extract_cookie(request, cookie_name, missing = NULL)

Arguments

request A shiny request object.
cookie_name The name of the cookie. Can contain any US-ASCII characters except for: the control character, space, a tab, or separator characters like ( ) < > , ; : / [ ] ? = { }.
missing The value to return if the requested cookie is not stored in the request. Defaults to NULL.

Value
The contents of that cookie.

Examples

req <- list(HTTP_COOKIE = "cookie1=expected_value; cookie2=1; cookie3=2")
extract_cookie(req, "cookie1")
extract_cookie(req, "cookie2")
extract_cookie(list(), "cookie1")
extract_cookie(NULL, "cookie1")
extract_cookies

Extract all cookies from a shiny request

Description

The shiny request object includes any cookies that are available to the app. This function extracts those cookies as a named list.

Usage

extract_cookies(request)

Arguments

request A shiny request object.

Value

All cookies in the request, as a list.

Examples

req <- list(HTTP_COOKIE = "cookie1=expected_value; cookie2=1; cookie3=2")
extract_cookies(req)
extract_cookies(list())
extract_cookies(NULL)

get_cookie

Read a cookie

Description

Read a cookie from the input object.

Usage

get_cookie(
  cookie_name,
  missing = NULL,
  session = shiny::getDefaultReactiveDomain()
)
**remove_cookie**

**Arguments**

- **cookie_name**: The name of the cookie. Can contain any US-ASCII characters except for: the control character, space, a tab, or separator characters like ( ) < > @ , ; : " / [ ] ? = { }.
- **missing**: The value to return if the requested cookie does not exist. Defaults to NULL.
- **session**: Shiny session in which the cookies can be found (the default should probably always be used).

**Value**

A character with the value of the cookie.

**Examples**

```r
server <- function(input, output, session) {
  get_cookie("my_cookie")
}
```

**Description**

Instruct the user’s browser to remove a cookie via JavaScript.

**Usage**

```
remove_cookie(cookie_name, session = shiny::getDefaultReactiveDomain())
```

**Arguments**

- **cookie_name**: The name of the cookie. Can contain any US-ASCII characters except for: the control character, space, a tab, or separator characters like ( ) < > @ , ; : " / [ ] ? = { }.
- **session**: Shiny session in which the cookies can be found (the default should probably always be used).

**Value**

A call to `session$sendCustomMessage()` which removes the targeted cookie.

**Examples**

```r
server <- function(input, output, server) {
  shiny::observeEvent(
    input$button_that_removes_cookie,
    remove_cookie("my_cookie")
  )
}
```
**set_cookie**

Create or update a cookie

---

**Description**

Instruct the user’s browser to create a cookie via JavaScript.

**Usage**

```r
set_cookie(
  cookie_name,
  cookie_value,
  expiration = 90,
  secure_only = NULL,
  domain = NULL,
  path = NULL,
  same_site = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

**Arguments**

- **cookie_name**  
The name of the cookie. Can contain any US-ASCII characters except for: the control character, space, a tab, or separator characters like ( ) < > , ; : " / .

- **cookie_value**  
The contents of the cookie as a single character value.

- **expiration**  
Days after which the cookie should expire. To remove an HttpOnly cookie, send a negative value for this attribute.

- **secure_only**  
Logical indicating whether the cookie should only be accessible via secure (https:) requests (except on localhost).

- **domain**  
The host to which the cookie will be sent (including subdomains). If this is NULL (default) the cookie will only be sent to the host of the page where this cookie was set (not including subdomains).

- **path**  
The path that must exist in the requested URL for the browser to send this cookie. Includes subdirectories.

- **same_site**  
One of "strict", "lax" (default), or "none", indicating when the cookie should be sent. When same_site = "none", secure_only must be TRUE.

- **session**  
Shiny session in which the cookies can be found (the default should probably always be used).

**Value**

A call to `session$sendCustomMessage()` which sets the targeted cookie.
Examples

```r
server <- function(input, output, server) {
  shiny::observeEvent(
    input$button_that_sets_cookie,
    set-cookie(
      "my_cookie",
      "the value of this cookie"
    )
  )
}
```

**set_cookie_on_load**  
*Shiny tag to add cookies on page load*

**Description**

Generate a `shiny::tagList()` which uses JavaScript to set a cookie in the user’s browser when the shiny app loads.

**Usage**

```r
set_cookie_on_load(
  cookie_name,  
  cookie_value,  
  expiration = 90,  
  secure_only = NULL,  
  domain = NULL,  
  path = NULL,  
  same_site = NULL
)
```

**Arguments**

- **cookie_name**  
The name of the cookie. Can contain any US-ASCII characters except for: the control character, space, a tab, or separator characters like ( ) < > @ : ; \ / . 

- **cookie_value**  
The contents of the cookie as a single character value.

- **expiration**  
Days after which the cookie should expire. To remove an HttpOnly cookie, send a negative value for this attribute.

- **secure_only**  
Logical indicating whether the cookie should only be accessible via secure (https:) requests (except on localhost).

- **domain**  
The host to which the cookie will be sent (including subdomains). If this is NULL (default) the cookie will only be sent to the host of the page where this cookie was set (not including subdomains).

- **path**  
The path that must exist in the requested URL for the browser to send this cookie. Includes subdirectories.

- **same_site**  
One of "strict", "lax" (default), or "none", indicating when the cookie should be sent. When same_site = "none", secure_only must be TRUE.
set_cookie_response

Value

A shiny::tagList() that provides the HTML and javascript to set the cookie.

Examples

```r
set_cookie_on_load("my_cookie", "contents of my cookie")
set_cookie_on_load("my_cookie", "contents of my cookie", expiration = 10)
```

--

set_cookie_response  Set cookie via HTTP header

Description

Send a shiny::httpResponse() that sets a cookie in the user’s browser. Note that this does not return a full shiny ui.

Usage

```r
set_cookie_response(
  cookie_name,
  cookie_value,
  expiration = 90,
  secure_only = NULL,
  domain = NULL,
  path = NULL,
  same_site = NULL,
  http_only = FALSE,
  redirect = NULL,
  ...
)
```

Arguments

- **cookie_name**  The name of the cookie. Can contain any US-ASCII characters except for: the control character, space, a tab, or separator characters like ( ) < > @ , ; : " / [ ] ? = { }.
- **cookie_value**  The contents of the cookie as a single character value.
- **expiration**  Days after which the cookie should expire. To remove an HttpOnly cookie, send a negative value for this attribute.
- **secure_only**  Logical indicating whether the cookie should only be accessible via secure (https:) requests (except on localhost).
- **domain**  The host to which the cookie will be sent (including subdomains). If this is NULL (default) the cookie will only be sent to the host of the page where this cookie was set (not including subdomains).
- **path**  The path that must exist in the requested URL for the browser to send this cookie. Includes subdirectories.
**set_cookie_response**

- **same_site**: One of "strict", "lax" (default), or "none", indicating when the cookie should be sent. When `same_site = "none"`, `secure_only` must be TRUE.

- **http_only**: Logical indicating whether the cookie should only be sent as part of an HTTP request. When this is FALSE (default), the cookie is accessible to JavaScript via the `Document.cookie` property.

- **redirect**: A relative or absolute URL where the user should be sent next. A typical case would be the same URL minus the query parameter that triggered the Set-cookie response.

- **...**: Additional parameters passed on to `shiny::httpResponse()`.

**Value**

A `shiny::httpResponse()` that sets the cookie.

**Examples**

```r
set_cookie_response("my_cookie", "contents of my cookie")
set_cookie_response("my_cookie", "contents of my cookie", expiration = 10)
set_cookie_response(
  "my_cookie", "contents of my cookie",
  content = "Your cookie is set."
)
set_cookie_response(
  "my_cookie", "contents of my cookie",
  redirect = "/"
)
```
Index

add_cookie_handlers, 2
cookie_dependency, 2
extract_cookie, 3
extract_cookies, 4
get_cookie, 4
htmltools::htmlDependency(), 3
remove_cookie, 5
set_cookie, 6
set_cookie_on_load, 7
set_cookie_response, 8
shiny::bootstrapPage(), 2
shiny::fluidPage(), 2
shiny::htmlTemplate(), 2
shiny::httpResponse(), 8, 9
shiny::tagList(), 2, 7, 8