Package: ambhtmx (via r-universe)

September 7, 2024

Title ambhtmx

Version 0.0.0.9002

Description Build a Full-stack R App with ambiorix and htmx.

License CC0

Encoding UTF-8

Language en

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1.9000

Depends R (>= 4.4)

Suggests dbplyr, devtools, ggplot2, here, pak, pkgdown (>= 2.1.0), plotly, scilis (>= 0.0.0.9000), signaculum (>= 0.0.0.9000), testthat, usethis, zeallot

Imports ambiorix (>= 2.1.0), b64, DBI, dplyr, glue, htmltools, htmlwidgets, pool, purrr, readr, rlang, RSQLite, stringr, tibble, uwu (>= 0.0.0.9000), withr

Remotes devOpifex/ambiorix, devOpifex/scilis, devOpifex/signaculum, JosiahParry/uwu

URL `https://jrosell.github.io/ambhtmx`, `https://github.com/jrosell/ambhtmx`

BugReports `https://github.com/jrosell/ambhtmx/issues`

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

RemoteUrl https://github.com/jrosell/ambhtmx

RemoteRef HEAD

RemoteSha 3e9113e428eeb0a169d86ba597ee095df7989518

Contents

---

ambhtmx *Creating an ambiorix + htmx app*

Description

Creating an ambiorix + htmx app

Usage

```
ambhtmx(
  dbname = NULL,
  value = tibble::tibble(),
  protocol = NULL,
  host = NULL,
  port = NULL,
  live = "",
  favicon = NULL,
  render_index = NULL,
  render_row = NULL
)
```

Arguments

dbname	file path to store a SQLite database (optional).
value	a 1 row tibble with the names and types of the columns (optional)
protocol	(default AMBHTMX_PROTOCOL or http)
host	(default AMBHTMX_HOST or 127.0.0.1)
port	(default AMBHTMX_PORT or 3000)
live	script with the file path (optional)

`favicon`	(optional)
`render_index`	function to be stored as a model method (optional)
`render_row`	function to be stored as a model method (optional)

Value

A list with the ambiorix app, the running context and the model methods.

---

amb_card *Generate component*

Description

Generate component

Usage

```
amb_card(
  ...,
  class = NULL,
  title = NULL,
  title_icon = NULL,
  title_class = NULL
)

amb_input_text(
  ...,
  id,
  label = NULL,
  value = "",
  input_class = NULL,
  hx_post = NULL
)

amb_button(..., class = "rounded-1", type = "button")

amb_htmlwidget(widget, ..., width = "100%", height = "400px")
```

Arguments

`...`	attributes to add to the container
`class`	customize
`title`	to customeize the title text
`title_icon`	to customize the title icon
`title_class`	to add more classes to the title

`id`	for the label and the input
`label`	customize
`value`	customize
`input_class`	customize
`hx_post`	customize
`type`	customize
`widget`	htmlwidget to convert as a shiny.tag
`width`	to customeize the width of the container
`height`	to customeize the width of the container

`process_error_get` *Process error get requests*

Description

Process error get requests

Usage

```
process_error_get(req, res, cookie_errors = "errors")
```

Arguments

`req`	request object
`res`	response object
`cookie_errors`	if you need to customize the name of the errors cookie

Value

the error character vector

`process_loggedin_middleware` *Process loggedin middleware*

Description

Process loggedin middleware

Usage

```
process_loggedin_middleware(
  req,
  res,
  user = Sys.getenv("AMBHTMX_USER"),
  cookie_loggedin = "loggedin"
)
```

Arguments

`req`	request object
`res`	response object
`user`	if you want to customize the required user or it uses AMBHTMX_USER
`cookie_loggedin`	if you need to customize the name of the loggedin cookie

Value

the updated request with the req$loggedin status

`process_login_get` *Render and send login page response*

Description

Render and send login page response

Usage

```
process_login_get(
  req,
  res,
  page_title = "Login",
  main = NULL,
  id = "login_form",
  login_url = "/login",
  style = "margin: 20px",
  cookie_errors = "errors"
)
```

Arguments

`req`	request object
`res`	response object
`page_title`	if you need to customize the title of the page
`main`	if you need to customize the body of the login page
`id`	if you need to customize the id of the login form
`login_url`	if you need to customize the url of the login form
`style`	if you need to customize the styles of the login form
`cookie_errors`	if you need to customize the name of the errors cookie

Value

the login page response

`process_login_post` *Process login requests*

Description

Process login requests

Usage

```
process_login_post(
  req,
  res,
  user_param = "user",
  password_param = "password",
  user = Sys.getenv("AMBHTMX_USER"),
  password = Sys.getenv("AMBHTMX_PASSWORD"),
  user_error = "Invalid user",
  password_error = "Invalid password",
  cookie_loggedin = "loggedin",
  cookie_errors = "errors",
  login_url = "/login",
  success_url = "/"
)
```

Arguments

`req`	request object
`res`	response object
`user_param`	if you need to customize the name of the user parameter
`password_param`	if you need to customize the name of the password parameter
`user`	if you want to customize the required user or it uses AMBHTMX_USER
`password`	if you want to customize the required password or it uses AMBHTMX_PASSWORD
`user_error`	if you need to customize the error message for the user
`password_error`	if you need to customize the error message for the password
`cookie_loggedin`	if you need to customize the name of the loggedin cookie
`cookie_errors`	if you need to customize the name of the errors cookie
`login_url`	if you need to customize the url of the login form
`success_url`	if you need to customize the url of the success loggedin process

Value

the login process response

`process_logout_get` *Process logout requests*

Description

Process logout requests

Usage

```
process_logout_get(req, res, cookie_loggedin = "loggedin", success_url = "/")
```

Arguments

`req`	request object
`res`	response object
`cookie_loggedin`	if you need to customize the name of the loggedin cookie
`success_url`	if you need to customize the url of the success loggedin process

Value

the logout process response

`render_page` *Render a custom page with a custom title and main content*

Description

Render a custom page with a custom title and main content

Usage

```
render_page(main = NULL, page_title = NULL, head_tags = NULL)
```

Arguments

`main`	htmltools object of the body of the html page
`page_title`	the title tag contents of the page
`head_tags`	optional htmltools object of the head of the html page

Details

It can throw exceptions, so handling exceptions is recommended, if not a must.

Value

the rendered html of the full html page with dependencies

`render_png` *Render a png image to a img tag*

Description

Render a png image to a img tag

Usage

```
render_png(p)
```

Arguments

`p`	a ggplot or another object that can be printed and captured as a png image

Value

img htmltools tag with a data encoded src attribute

`render_tags` *Render tags to character vector*

Description

Render tags to character vector

Usage

```
render_tags(...)
```

Arguments

`...`	one or more htmltools objects.

Value

a character representation of input

`script_from_js_tpl` *Generate script from js template*

Description

Generate script from js template

Usage

```
script_from_js_tpl(file, ...)
```

Arguments

`file`	path to a js file
`...`	mutiple named arguments with the value to replace

Examples

```
if (FALSE){
  # replaces {init} to 0
  script_from_js_tpl("script.js", init = "init")
}
```

`script_tpl` *Generate script from raw js template*

Description

Generate script from raw js template

Usage

```
script_tpl(raw_content, ...)
```

Arguments

`raw_content`	path to a js file
`...`	mutiple named arguments with the value to replace

`send_page` *Render a page and send the respose*

Description

Render a page and send the respose

Usage

```
send_page(main = NULL, res, ...)
```

Arguments

`main`	htmltools object of the body of the html page
`res`	response object
`...`	other paramters to the render page function

`send_tags` *Render a custom page with a custom title and main content*

Description

Render a custom page with a custom title and main content

Usage

```
send_tags(main = NULL, res, ...)
```

Arguments

`main`	htmltools object to render
`res`	response object
`...`	htmltools object to render

`style_from_css_tpl` *Generate style from css template*

Description

Generate style from css template

Usage

```
style_from_css_tpl(file, ...)
```

Arguments

`file`	path to a css file
`...`	mutiple named arguments with the value to replaces

Examples

```
if (FALSE){
  # replaces "var(--tpl-background)" to "red"
  style_from_css_tpl("styles.css", background = "red")
}
```

`style_tpl` *Generate style from raw css*

Description

Generate style from raw css

Usage

```
style_tpl(raw_content, ...)
```

Arguments

`raw_content`	contents with tpl in css
`...`	mutiple named arguments with the value to replaces

`tags` *Create HTML tags without requiring htmltools::tags$*

Description

Create an R object that represents an HTML tag. For convenience, common HTML tags (e.g., `<div>`) can be created by calling for their tag name directly (e.g., `div()`). To create less common HTML5 (or SVG) tags (e.g., `<article>`), use the `tags` list collection (e.g., `tags$article()`). To create other non HTML/SVG tags, use the lower-level `tag()` constructor.

Usage

```
button(..., .noWS = NULL, .renderHook = NULL)

textarea(..., .noWS = NULL, .renderHook = NULL)

input(..., .noWS = NULL, .renderHook = NULL)

label(..., .noWS = NULL, .renderHook = NULL)

nav(..., .noWS = NULL, .renderHook = NULL)

li(..., .noWS = NULL, .renderHook = NULL)

ul(..., .noWS = NULL, .renderHook = NULL)

ol(..., .noWS = NULL, .renderHook = NULL)

form(..., .noWS = NULL, .renderHook = NULL)
```

```
style(..., .noWS = NULL, .renderHook = NULL)

script(..., .noWS = NULL, .renderHook = NULL)
```

Arguments

`...`	Tag attributes (named arguments) and children (unnamed arguments). A named argument with an `NA` value is rendered as a boolean attributes (see example). Children may include any combination of: • Other tags objects • `HTML()` strings • `htmlDependency()`s • Single-element atomic vectors • `list()`s containing any combination of the above
`.noWS`	Character vector used to omit some of the whitespace that would normally be written around this tag. Valid options include `before`, `after`, `outside`, `after-begin`, and `before-end`. Any number of these options can be specified.
`.renderHook`	A function (or list of functions) to call when the `tag` is rendered. This function should have at least one argument (the `tag`) and return anything that can be converted into tags via `as.tags()`. Additional hooks may also be added to a particular `tag` via `tagAddRenderHook()`.

Value

A `list()` with a `shiny.tag` class that can be converted into an HTML string via `as.character()` and saved to a file with `save_html()`.

Index