

# Package: tower (via r-universe)

October 23, 2024

**Title** Easy Middle Ware Library for 'shiny'

**Version** 0.2.0

**Description** The best way to implement middle ware for 'shiny' Applications. 'tower' is designed to make implementing behavior on top of 'shiny' easy with a layering model for incoming HTTP requests and server sessions. 'tower' is a very minimal package with little overhead, it is mainly meant for other package developers to implement new behavior.

**URL** <https://github.com/ixpantia/tower>

**BugReports** <https://github.com/ixpantia/tower/issues>

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**Imports** compiler, glue, purrr, stringr, curl, jsonlite, rlang

**Suggests** testthat (>= 3.0.0), shiny

**Config/testthat/edition** 3

**Repository** <https://ixpantia.r-universe.dev>

**RemoteUrl** <https://github.com/ixpantia/tower>

**RemoteRef** HEAD

**RemoteSha** b6a83659138b4e6798672d1fbc2a5ee2ab2cb708

## Contents

add_body . . . . .	2
add_body_json . . . . .	3
add_cookie . . . . .	3
add_delete_route . . . . .	4
add_get_route . . . . .	4
add_http_layer . . . . .	5

add_patch_route . . . . .	5
add_post_route . . . . .	6
add_put_route . . . . .	6
add_route . . . . .	7
add_server_layer . . . . .	7
app_into_parts . . . . .	8
build_http_cookie . . . . .	8
build_response . . . . .	9
build_tower . . . . .	9
create_tower . . . . .	10
print.tower . . . . .	10
req_body_form . . . . .	11
req_body_json . . . . .	11
req_cookies . . . . .	12
req_query . . . . .	12
response_builder . . . . .	13
set_content_type . . . . .	13
set_header . . . . .	14
set_status . . . . .	14

<b>Index</b>	<b>15</b>
--------------	-----------

---

add_body	<i>Add a body to a response</i>
----------	---------------------------------

---

### Description

Adds a body to a response, if no content type is set, it will be detected

### Usage

```
add_body(res, body)
```

### Arguments

res	A response builder object
body	The body to add

### Value

The response builder object

---

add_body_json	<i>Add a body to a response as JSON</i>
---------------	---

---

**Description**

Adds a body to a response as JSON

**Usage**

```
add_body_json(res, body)
```

**Arguments**

res	A response builder object
body	The body to add

**Value**

The response builder object

---

add_cookie	<i>Add a cookie to a response</i>
------------	-----------------------------------

---

**Description**

Adds a cookie to a response

**Usage**

```
add_cookie(res, name, value)
```

**Arguments**

res	A response builder object
name	The name of the cookie
value	The value of the cookie

**Value**

The response builder object

---

add_delete_route	<i>Add a DELETE route</i>
------------------	---------------------------

---

**Description**

Adds a DELETE route to a tower

**Usage**

```
add_delete_route(tower, path, handler)
```

**Arguments**

tower	A tower object
path	A string containing the path to match
handler	A function to call when the route is matched

**Value**

The tower with the added DELETE route

---

add_get_route	<i>Add a GET route</i>
---------------	------------------------

---

**Description**

Adds a GET route to a tower

**Usage**

```
add_get_route(tower, path, handler)
```

**Arguments**

tower	A tower object
path	A string containing the path to match
handler	A function to call when the route is matched

**Value**

The tower with the added GET route

---

add_http_layer	<i>Add an HTTP layer to a tower</i>
----------------	-------------------------------------

---

**Description**

Add an HTTP layer to a tower. This layer will be called before the 'shiny' app's httpHandler.

**Usage**

```
add_http_layer(tower, layer)
```

**Arguments**

tower	A tower
layer	A function that takes a request and returns either a response. A layer can short circuit by returning a response directly or call the next layer will req\$NEXT(req) which will call the next layer in the middleware.

**Value**

The tower with the added layer

---

add_patch_route	<i>Add a PATCH route</i>
-----------------	--------------------------

---

**Description**

Adds a PATCH route to a tower

**Usage**

```
add_patch_route(tower, path, handler)
```

**Arguments**

tower	A tower object
path	A string containing the path to match
handler	A function to call when the route is matched

**Value**

The tower with the added PATCH route

---

add_post_route	<i>Add a POST route</i>
----------------	-------------------------

---

**Description**

Adds a POST route to a tower

**Usage**

```
add_post_route(tower, path, handler)
```

**Arguments**

tower	A tower object
path	A string containing the path to match
handler	A function to call when the route is matched

**Value**

The tower with the added POST route

---

add_put_route	<i>Add a PUT route</i>
---------------	------------------------

---

**Description**

Adds a PUT route to a tower

**Usage**

```
add_put_route(tower, path, handler)
```

**Arguments**

tower	A tower object
path	A string containing the path to match
handler	A function to call when the route is matched

**Value**

The tower with the added PUT route

---

add_route	<i>Add an HTTP layer to a tower</i>
-----------	-------------------------------------

---

**Description**

Adds an HTTP layer to a tower

**Usage**

```
add_route(tower, method = "GET", path, handler)
```

**Arguments**

tower	A tower object
method	A string containing the HTTP method to match
path	A string containing the path to match
handler	A function to call when the layer is matched

**Value**

The tower with the added route

---

add_server_layer	<i>Add a server layer to a tower</i>
------------------	--------------------------------------

---

**Description**

Add a server layer to a tower. This layer will run before the 'shiny' app's server function. This is useful for adding custom logic to the server function without modifying the original server function.

**Usage**

```
add_server_layer(tower, layer)
```

**Arguments**

tower	A tower
layer	A function that takes input, output, and session and has no return value. This function will be called before the original server function. If you want to short-circuit the server use an exception.

**Value**

The tower with the added layer

---

app_into_parts	<i>Into parts</i>
----------------	-------------------

---

**Description**

Splits a shiny.appobj into its parts, the ui and server

**Usage**

```
app_into_parts(app)
```

**Arguments**

app	A shiny.appobj
-----	----------------

**Value**

A list with the ui and server handlers

---

build_http_cookie	<i>Build a cookie</i>
-------------------	-----------------------

---

**Description**

Builds an HttpOnly cookie from a key and value

**Usage**

```
build_http_cookie(key, value)
```

**Arguments**

key	A string containing the cookie key
value	A string containing the cookie value

**Value**

A string containing the formatted cookie



---

`build_response`      *Build a response*

---

**Description**

Builds a response

**Usage**

```
build_response(res)
```

**Arguments**

`res`              A response builder object

**Value**

A 'shiny' response object

---

`build_tower`      *Build a 'shiny' app from a tower*

---

**Description**

Build a 'shiny' app from a tower. This will create a new 'shiny' app with the specified layers added.

**Usage**

```
build_tower(tower)
```

**Arguments**

`tower`            A tower

**Value**

A 'shiny' app object that can be started

---

create_tower	<i>Create a new tower</i>
--------------	---------------------------

---

**Description**

Create a new tower to build upon.

**Usage**

```
create_tower(app)
```

**Arguments**

app	A 'shiny' app object
-----	----------------------

**Value**

A new tower object to add more layers to

---

print.tower	<i>Print a tower</i>
-------------	----------------------

---

**Description**

Print a tower

**Usage**

```
## S3 method for class 'tower'  
print(x, ...)
```

**Arguments**

x	A tower
...	Ignored arguments (for compatibility with print)

**Value**

No return value, called for side effects

---

req_body_form	<i>Extract form data from a request</i>
---------------	---

---

**Description**

Extracts form data from a request

**Usage**

```
req_body_form(req)
```

**Arguments**

req	A request object
-----	------------------

**Value**

A list containing the form data in the body

---

req_body_json	<i>Extract the request body from a JSON request</i>
---------------	---

---

**Description**

Extracts the request body from a JSON request

**Usage**

```
req_body_json(req, ...)
```

**Arguments**

req	A request object
...	Additional arguments to pass to <a href="#">fromJSON</a> when parsing the request body. This will only be used the first time the request body is parsed. Subsequent calls will return the cached result.

**Value**

The R object representation of the body's JSON content

---

req_cookies	<i>Extract cookies from a request</i>
-------------	---------------------------------------

---

**Description**

Extracts cookies from a request

**Usage**

```
req_cookies(req)
```

**Arguments**

req	A request object
-----	------------------

**Value**

A list containing the cookies

---

req_query	<i>Extract query parameters from a request</i>
-----------	--

---

**Description**

Extracts query parameters from a request

**Usage**

```
req_query(req)
```

**Arguments**

req	A request object
-----	------------------

**Value**

A list containing the query parameters

---

response_builder	<i>Create a response builder</i>
------------------	----------------------------------

---

**Description**

Creates a response builder

**Usage**

```
response_builder()
```

**Value**

A response builder object

---

set_content_type	<i>Set the content type of a response</i>
------------------	---

---

**Description**

Sets the content type of a response

**Usage**

```
set_content_type(res, content_type)
```

**Arguments**

res	A response builder object
content_type	The content type to set

**Value**

The response builder object

---

set_header	<i>Set a header on a response</i>
------------	-----------------------------------

---

**Description**

Sets or adds a header to a response

**Usage**

```
set_header(res, name, value)
```

**Arguments**

res	A response builder object
name	The name of the header
value	The value of the header

**Value**

The response builder object

---

set_status	<i>Set the status of a response</i>
------------	-------------------------------------

---

**Description**

Sets the status of a response

**Usage**

```
set_status(res, status)
```

**Arguments**

res	A response builder object
status	The status to set

**Value**

The response builder object

# Index

`add_body`, 2  
`add_body_json`, 3  
`add_cookie`, 3  
`add_delete_route`, 4  
`add_get_route`, 4  
`add_http_layer`, 5  
`add_patch_route`, 5  
`add_post_route`, 6  
`add_put_route`, 6  
`add_route`, 7  
`add_server_layer`, 7  
`app_into_parts`, 8  
  
`build_http_cookie`, 8  
`build_response`, 9  
`build_tower`, 9  
  
`create_tower`, 10  
  
`fromJSON`, 11  
  
`print.tower`, 10  
  
`req_body_form`, 11  
`req_body_json`, 11  
`req_cookies`, 12  
`req_query`, 12  
`response_builder`, 13  
  
`set_content_type`, 13  
`set_header`, 14  
`set_status`, 14