

Package: mighty.component (via r-universe)

October 16, 2026

Title Mighty Components for ADaM Generation

Version 0.1.0.9009

Description Reusable code templates ('components') for the 'mighty' framework, used to generate Analysis Data Model (ADaM) programs. Components are 'Mustache' templates documented with 'roxygen2'-like tags. They can be retrieved from local directories, 'GitHub' or URLs, rendered with study-specific parameters, validated, evaluated, and unit tested with code coverage.

License Apache License (>= 2)

URL <https://novonordisk-opensource.github.io/mighty.component>,
<https://github.com/NovoNordisk-OpenSource/mighty.component>

BugReports <https://github.com/NovoNordisk-OpenSource/mighty.component/issues>

Depends R (>= 4.1)

Imports cli, glue, R6 (>= 2.4.0), rlang, S7, whisker, withr (>= 2.5.0), xml2, xmlparsedata, zephyr

Suggests admiral, callr, covr, curl, dplyr, gh, httr2, knitr, pharmaverseadam, remotes, rmarkdown, testthat (>= 3.2.0), tibble, tidyr

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Language en-US

Config/pak/sysreqs libxml2-dev

Repository <https://novonordisk-opensource.r-universe.dev>

Date/Publication 2026-10-12 13:33:14 UTC

RemoteUrl <https://github.com/NovoNordisk-OpenSource/mighty.component>

RemoteRef HEAD

RemoteSha ccf9a7ff6002c71c6e8e5c8c2036957f8579bd25

Contents

find_component	2
get_component	3
get_test_component	4
list_components	5
mighty.component-options	6
mighty_component	7
mighty_component_rendered	11
mighty_component_test	13
mighty_repo	16
mighty_repo_github	17
mighty_repo_local	18
mighty_repo_url	18
mighty_repos	19
Index	21

find_component	<i>Find a component</i>
----------------	-------------------------

Description

Find a component in one or more repos. Returns NULL if no repo contains it, while `get_component()` raises an error. Local repo directories must exist, otherwise an error is raised.

Usage

```
find_component(component, repos = NULL)
```

Arguments

- | | |
|-----------|---|
| component | character(1) Component name. If repos is NULL, path to a component file. |
| repos | Where to look. One of: <ul style="list-style-type: none">• NULL (default): component is a file path.• character vector of repo specs, in priority order. See <code>mighty_repo()</code>.• A <code>mighty_repo_class</code> object.• A list of repo specs or <code>mighty_repo_class</code> objects, in priority order.• A <code>mighty_repos()</code> collection. Character vectors and lists are converted once with <code>mighty_repos()</code> . |

Value

A `mighty_component` object, or NULL if no repo contains the component.

See Also

```
get_component(), list_components()
```

Examples

```
path <- system.file("examples", package = "mighty.component")
find_component("ady", repos = path)

find_component("does_not_exist", repos = path)
```

get_component	<i>Retrieve a component</i>
---------------	-----------------------------

Description

- `get_component()`: Find a component. Raises an error if it is not found.
- `get_rendered_component()`: Find a component and render it with params.

Usage

```
get_component(component, repos = NULL)

get_rendered_component(component, params = list(), repos = NULL)
```

Arguments

component	character(1) Component name. If repos is NULL, path to a component file.
repos	Where to look. One of: • NULL (default): component is a file path. • character vector of repo specs, in priority order. See mighty_repo(). • A <code>mighty_repo_class</code> object. • A list of repo specs or <code>mighty_repo_class</code> objects, in priority order. • A mighty_repos() collection. Character vectors and lists are converted once with mighty_repos() .
params	named list of parameters passed to <code>\$render()</code> . See the component's <code>@param</code> tags (<code>component\$params</code>).

Details

Components are `.mustache` or `.R` files. Both use the tags described in [mighty_component](#).

- `.mustache` files are Mustache templates.
- `.R` files are plain R components. They cannot have `@param` tags or Mustache placeholders.

Value

- `get_component()`: A [mighty_component](#) object.
- `get_rendered_component()`: A [mighty_component_rendered](#) object.

See Also

[find_component\(\)](#), [mighty_component](#), [mighty_component_rendered](#)

Examples

```
path <- system.file("examples", "ady.mustache", package = "mighty.component")
get_component(path)

get_rendered_component(
  component = path,
  params = list(domain = "ADAE", variable = "ASTDY", date = "ASTDT")
)

# Find by name in a repo
repo <- system.file("examples", package = "mighty.component")
get_component("ady", repos = repo)
```

get_test_component	Create a test component
--------------------	-------------------------

Description

Retrieve and render a component as a [mighty_component_test](#) object, for unit tests with code coverage. See [mighty_component_test](#) for the workflow.

Requires the callr and covr packages.

Usage

```
get_test_component(
  component,
  params = list(),
  repos = NULL,
  check_coverage = TRUE,
  teardown_env = parent.frame()
)
```

Arguments

component	character(1) Component name. If repos is NULL, path to a component file.
params	named list of parameters passed to <code>\$render()</code> . See the component's <code>@param</code> tags (<code>component\$params</code>).
repos	Where to look. One of: • NULL (default): component is a file path. • character vector of repo specs, in priority order. See mighty_repo(). • A <code>mighty_repo_class</code> object.

- A list of repo specs or `mighty_repo_class` objects, in priority order.
- A `mighty_repos()` collection.

Character vectors and lists are converted once with `mighty_repos()`.

`check_coverage` `logical(1)` If TRUE (default), `$check_coverage()` runs when `teardown_env` ends. It raises an error if any line has not run.

`teardown_env` Environment that controls when the coverage check runs. Defaults to the calling environment, e.g. the `test_that()` block.

Value

A `mighty_component_test` object.

See Also

`get_rendered_component()`, `mighty_component_test`

Examples

```
path <- system.file("examples", "ady.mustache", package = "mighty.component")
x <- get_test_component(
  component = path,
  params = list(domain = "adae", variable = "ASTDY", date = "ASTDT"),
  check_coverage = FALSE
)

adae <- data.frame(
  TRTSDT = as.Date("2024-01-01"),
  ASTDT = as.Date(c("2024-01-01", "2024-01-10"))
)

x$assign("adae", adae)$eval()
x$get("adae")
x$percent_coverage
x$close()
```

<code>list_components</code>	<i>List components in repos</i>
------------------------------	---------------------------------

Description

List the components in one or more repos. See `mighty_repo()` for the component layout. Files starting with `test-` are not listed.

Usage

```
list_components(repos, as = c("character", "list", "tibble"))
```

Arguments

- repos** Where to look. One of:
- character vector of repo specs. See [mighty_repo\(\)](#).
 - A `mighty_repo_class` object.
 - A list of repo specs or `mighty_repo_class` objects.
 - A [mighty_repos\(\)](#) collection.
- Character vectors and lists are converted once with [mighty_repos\(\)](#).
- as** Format to list the components in:
- "character": component names (filenames without extension).
 - "list": metadata for each component.
 - "tibble": metadata for each component as a tibble. Requires the `tibble` and `tidyr` packages.
- When a component exists in several repos, the metadata is taken from the first repo, as in [find_component\(\)](#).

Value

Depending on as:

- "character": character vector of unique component names.
- "list": list with one element per component, each a named list with id, title, description, type, origin, method, params, depends, outputs and code.
- "tibble": tibble with one row per component and the same columns.

See Also

[get_component\(\)](#), [find_component\(\)](#)

Examples

```
path <- system.file("examples", package = "mighty.component")
list_components(path)

list_components(path, as = "list") |>
  str(max.level = 2)
```

Description

verbosity_level:

Verbosity level for functions in mighty.component. See [zephyr::verbosity_level](#) for details.

- Default: NA_character_
- Option: mighty.component.verbosity_level
- Environment: R_MIGHTY.COMPONENT_VERBOSITY_LEVEL

max_tries:

Maximum number of attempts for GitHub and URL requests. See [mighty_repo_github\(\)](#) and [mighty_repo_url\(\)](#) for which errors are retried.

- Default: 3L
- Option: mighty.component.max_tries
- Environment: R_MIGHTY.COMPONENT_MAX_TRIES

timeout:

Maximum number of seconds per URL request attempt. See [mighty_repo_url\(\)](#).

- Default: 5
- Option: mighty.component.timeout
- Environment: R_MIGHTY.COMPONENT_TIMEOUT

mighty_component

Mighty component class

Description

R6 class for a component.

A component is a code template. Its code takes an input data set and returns a modified version with new or changed columns or rows. Components are documented with roxygen-like tags.

Use [get_component\(\)](#) to create a component from a file or repo.

Details

Templates:

A template starts with tags in roxygen comments (#'). All lines below the @code tag are the R code.

Templates use **Mustache** placeholders. `$render()` fills them in with `whisker::whisker.render()`.

- `{{{name}}}` inserts the value of parameter name. Always use triple braces. Double braces (`{{name}}`) HTML-escape the value, e.g. `a<b` becomes `a<b`;
- `{{#name}}...{{/name}}` repeats the enclosed text for each element of the vector name. Use `{{{.}}}` to insert the current element.

Tags:

A tag continues until the next tag. Required tags must appear exactly once.

Tag	Required	Description	Example
@title	Yes	Title of the component.	@title Double a column
@description	Yes	Description of the component.	@description Creates a new column.
@param	No	Name, then description. One per placeholder.	@param domain 'character' Name of the
@type	Yes	Component type. See <i>Types</i> below.	@type column
@origin	Yes	CDISC origin. See allowed values below.	@origin Derived
@method	Yes	Free-text method description for define.xml.	@method Two times the input column
@depends	No	Input domain, then column. Repeat for each.	@depends {{{domain}}} USUBJID
@outputs	No	Column created. Repeat for each.	@outputs {{{output}}}
@code	Yes	Last tag. All lines below are the code.	@code

@origin must be one of Assigned, Collected, Derived, Not Available, Other, Predecessor, Protocol.

@depends is split at the first space into domain and column. Do not use spaces inside its placeholders, e.g. use {{{domain}}}, not {{ domain }}.

Types:

- column: Adds or modifies columns. The row count is unchanged.
- row: Adds, removes or modifies rows.
- parameter: Derives a new PARAMCD (BDS parameter).
- internal: Helper step with no define.xml output.

Conventions:

1. The input data set is {{{domain}}}.
2. The code assigns the result back to {{{domain}}}.
3. Every placeholder is declared with @param.
4. Functions are called with explicit namespaces, e.g. dplyr::mutate().
5. Joins specify by. This is enforced when rendering.

Validation:

When a component is rendered, the code is parsed and must be valid R. Joins from dplyr, tidylog and dbplyr without a by argument raise an error. Column existence is not checked.

Example:

This template creates a new column output as two times the existing column input:

```
#' @title Double a column
#' @description
#' Creates a new column as two times an existing column.
#'
#' @param domain `character` Name of the domain
#' @param output `character` Name of the new column
#' @param input `character` Name of the existing column
#' @type column
```

```
#' @origin Derived
#' @method Two times the input column
#' @depends {{{domain}}} {{{input}}}
#' @outputs {{{output}}}
#' @code
{{{domain}}} <- {{{domain}}} |>
  dplyr::mutate(
    {{{output}}} = 2 * {{{input}}}
  )
```

Rendered with domain = "ADSL", output = "A" and input = "B", the code (\$code) is:

```
ADSL <- ADSL |>
  dplyr::mutate(
    A = 2 * B
  )
```

The tags are rendered too, e.g. \$depends has domain ADSL and column B.

Active bindings

id character(1) Component name without file extension.
 title character(1) Title of the component.
 description character(1) Description of the component.
 code character Lines below @code.
 template character Full template.
 type character(1) Component type. One of column, row, parameter, internal.
 origin character(1) CDISC origin. One of Assigned, Collected, Derived, Not Available, Other, Predecessor, Protocol.
 method character(1) Method description for define.xml.
 depends data.frame with columns domain and column.
 outputs character Columns created by the component.
 params data.frame with columns name and description.

Methods

Public methods:

- `mighty_component$new()`
- `mighty_component$print()`
- `mighty_component$render()`
- `mighty_component$document()`
- `mighty_component$clone()`

`mighty_component$new()`: Create a component from a template.

Usage:

```
mighty_component$new(template, id)
```

Arguments:

template character Template, one element per line. See Details.

id character(1) Component ID.

mighty_component\$print(): Print the component.

Usage:

```
mighty_component$print()
```

Returns: (invisible) self

mighty_component\$render(): Render the component with [whisker::whisker.render\(\)](#).

Usage:

```
mighty_component$render(...)
```

Arguments:

... Named parameters used to render the template. Supply one for each @param tag and no others.

For @type row components, domain can be a subset marker: ".mighty_subset(<domain>, '<subset>')", where <subset> is an R expression in a string, e.g. domain = ".mighty_subset(ADLB, 'PARAMCD =

The code then only processes the rows where <subset> is TRUE. Other rows are kept unchanged. The marker is not allowed for other parameters or types.

Returns: Object of class [mighty_component_rendered](#)

mighty_component\$document(): Create documentation in markdown format. Requires the knitr package.

Usage:

```
mighty_component$document()
```

Returns: (invisible) character(1) Markdown documentation. Also printed to the console.

mighty_component\$clone(): The objects of this class are cloneable with this method.

Usage:

```
mighty_component$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

[get_component\(\)](#), [mighty_component_rendered](#), [vignette\("mighty-component"\)](#)

Examples

```
path <- system.file("examples", "ady.mustache", package = "mighty.component")
x <- mighty_component$new(template = readLines(path), id = "ady")
x
```

```
x$params
x$depends
```

```
x$render(domain = "ADAE", variable = "ASTDY", date = "ASTDT")
```

```
x$document()
```

```
mighty_component_rendered
```

Rendered mighty component class

Description

R6 class for a rendered component, created by [mighty_component\\$render\(\)](#) or [get_rendered_component\(\)](#).

A rendered component can be:

- Streamed into an R script with `$stream()`.
- Evaluated in an environment with `$eval()`.

Super class

[mighty_component](#) -> mighty_component_rendered

Methods

Public methods:

- [mighty_component_rendered\\$new\(\)](#)
- [mighty_component_rendered\\$print\(\)](#)
- [mighty_component_rendered\\$stream\(\)](#)
- [mighty_component_rendered\\$eval\(\)](#)
- [mighty_component_rendered\\$clone\(\)](#)

[mighty_component_rendered\\$new\(\)](#): Create a rendered component from a rendered template. The code is validated. See the Validation section in [mighty_component](#).

Usage:

```
mighty_component_rendered$new(template, id)
```

Arguments:

template character Rendered template, one element per line.

id character(1) Component ID.

[mighty_component_rendered\\$print\(\)](#): Print the rendered component, including its code.

Usage:

```
mighty_component_rendered$print()
```

Returns: (invisible) self

`mighty_component_rendered$stream()`: Write the code to an R script. The code is appended if the file exists.

Usage:

```
mighty_component_rendered$stream(path)
```

Arguments:

`path` `character(1)` Path to the R script.

Returns: (invisible) self

`mighty_component_rendered$eval()`: Evaluate the code.

Usage:

```
mighty_component_rendered$eval(envir = parent.frame())
```

Arguments:

`envir` Environment to evaluate the code in. Defaults to the calling environment.

Returns: Value of the last evaluated expression of the code.

`mighty_component_rendered$clone()`: The objects of this class are cloneable with this method.

Usage:

```
mighty_component_rendered$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

[get_rendered_component\(\)](#), [mighty_component](#)

Examples

```
path <- system.file("examples", "ady.mustache", package = "mighty.component")
x <- get_component(path)$render(
  domain = "adae",
  variable = "ASTDY",
  date = "ASTDT"
)
x$code
```

```
# Write the code to an R script
script <- tempfile(fileext = ".R")
x$stream(script)
readLines(script)
unlink(script)
```

```
# Evaluate the code in a new environment
env <- new.env()
env$adae <- data.frame(
  TRTSDT = as.Date("2024-01-01"),
  ASTDT = as.Date(c("2024-01-01", "2024-01-10"))
)
```

```
)
x$eval(envir = env)
env$adae
```

mighty_component_test *Test mighty component class*

Description

R6 class for unit testing a component with code coverage tracking. The code runs in a separate R session, and the lines executed are counted.

Details

Use `get_test_component()` to create a test component. The workflow is:

1. Create the test component with `get_test_component()`.
2. Assign input data with `$assign()`.
3. Run the code and update coverage with `$eval()`.
4. Retrieve results with `$get()`.
5. Test the results with `testthat expect_*()` functions.
6. Close the session with `$close()`.

By default, `get_test_component()` checks coverage with `$check_coverage()` when the test ends. Coverage is kept in the current R session, so it can still be checked after `$close()`.

The code runs inside a function, with `<-` replaced by `<<-`, so assignments are made in the session's global environment.

Super classes

`mighty_component` -> `mighty_component_rendered` -> `mighty_component_test`

Active bindings

`percent_coverage` `numeric(1)` Percentage of lines covered (0-100).

`line_coverage` `data.frame` with columns `line` and `value` (number of times the line has run).

Methods

Public methods:

- `mighty_component_test$new()`
- `mighty_component_test$print()`
- `mighty_component_test$assign()`
- `mighty_component_test$get()`

- `mighty_component_test$ls()`
- `mighty_component_test$eval()`
- `mighty_component_test$check_coverage()`
- `mighty_component_test$close()`
- `mighty_component_test$clone()`

`mighty_component_test$new()`: Create a test component from a rendered template. Starts a new R session. Requires the callr and covr packages.

Usage:

`mighty_component_test$new(template, id)`

Arguments:

`template` character Rendered template, one element per line.

`id` character(1) Component ID.

`mighty_component_test$print()`: Print the test component with its code coverage.

Usage:

`mighty_component_test$print()`

Returns: (invisible) self

`mighty_component_test$assign()`: Assign a variable in the test session.

Usage:

`mighty_component_test$assign(x, value)`

Arguments:

`x` character(1) Variable name.

`value` Value to assign.

Returns: (invisible) self

`mighty_component_test$get()`: Get a variable from the test session.

Usage:

`mighty_component_test$get(x)`

Arguments:

`x` character(1) Variable name.

Returns: Value of the variable.

`mighty_component_test$ls()`: List the variables in the test session.

Usage:

`mighty_component_test$ls()`

Returns: character Variable names.

`mighty_component_test$eval()`: Run the code in the test session and update the coverage.

Usage:

`mighty_component_test$eval()`

Returns: (invisible) self

`mighty_component_test$check_coverage()`: Check that every line of the code has run at least once. Raises an error otherwise.

Usage:

`mighty_component_test$check_coverage()`

Returns: (invisible) self

`mighty_component_test$close()`: Close the test session. Called automatically when the object is garbage collected.

Usage:

`mighty_component_test$close()`

Returns: (invisible) self

`mighty_component_test$clone()`: The objects of this class are cloneable with this method.

Usage:

`mighty_component_test$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

[get_test_component\(\)](#)

Examples

```
path <- system.file("examples", "ady.mustache", package = "mighty.component")
x <- get_test_component(
  component = path,
  params = list(domain = "adae", variable = "ADY", date = "ADT"),
  check_coverage = FALSE
)
x$percent_coverage

x$assign(
  "adae",
  data.frame(TRTSDT = as.Date("2024-01-01"), ADT = as.Date("2024-01-05"))
)
x$eval()
x$ls()
x$get("adae")$ADY
x$percent_coverage
x$check_coverage()
x$close()
```

mighty_repo

*Component repos***Description**

- `mighty_repo()`: Create a component repo from a spec.
- `mighty_repo_class`: Abstract parent class of all component repos. It has no properties and cannot be created directly.

Specs have the form `type::path`. Supported types:

- `local`: A local directory, e.g. `local::inst/examples`. A spec without a prefix is treated as `local`.
- `github`: A GitHub repository, e.g. `github::owner/repo/subdir@ref`. See [mighty_repo_github\(\)](#).
- `url`: Raw files under a base URL, e.g. `url::https://example.com/components`. See [mighty_repo_url\(\)](#).

All repo types share one layout. A component name is the file `<name>.R` or `<name>.mustache`, either directly in the repo or in a directory named after the component (`<name>/<name>.R`). A name with an extension only matches that file.

Usage

```
mighty_repo(spec)
```

Arguments

`spec` `character(1)` repo spec. See description.

Value

`mighty_repo()`: An object inheriting from `mighty_repo_class`.

See Also

[mighty_repo_local\(\)](#), [mighty_repo_github\(\)](#), [mighty_repo_url\(\)](#), [mighty_repos\(\)](#)

Examples

```
path <- system.file("examples", package = "mighty.component")
mighty_repo(path)
mighty_repo(paste0("local::", path))
```

mighty_repo_github	<i>GitHub component repo</i>
--------------------	------------------------------

Description

A component repo hosted on GitHub. When the object is created, ref is resolved to a commit SHA and the repository tarball for that commit is downloaded and extracted to a temporary directory. Resolved SHAs are cached per ref and tarballs per commit for the session, so all lookups afterwards are local. Restart the session to pick up new commits on a branch.

spec is a remotes-style repository reference: owner/repo, owner/repo/subdir, owner/repo@ref or a combination.

Transient API errors (HTTP 5xx and network failures) are retried. The number of attempts is set by the max_tries option. See [mighty.component-options](#).

Requires the gh and remotes packages. Private repositories need a GitHub token. gh reads it from the GITHUB_PAT or GITHUB_TOKEN environment variable, or the Git credential store. See [gh::gh_token\(\)](#).

Usage

```
mighty_repo_github(spec)
```

Arguments

spec	character(1) GitHub repository reference. See description.
------	--

Value

A mighty_repo_github object. It inherits from [mighty_repo_local\(\)](#), with path set to the downloaded repository or its subdir.

See Also

[mighty_repo\(\)](#)

Examples

```
repo <- mighty_repo_github(  
  spec = "NovoNordisk-OpenSource/mighty.standards/components@main"  
)  
find_component(component = "dummy", repos = repo)
```

mighty_repo_local	<i>Local component repo</i>
-------------------	-----------------------------

Description

A component repo in a local directory. See [mighty_repo\(\)](#) for the component layout. An error is raised if more than one file matches a component.

Usage

```
mighty_repo_local(path)
```

Arguments

path character(1) Path to an existing directory.

Value

A mighty_repo_local object.

Examples

```
path <- system.file("examples", package = "mighty.component")
mighty_repo_local(path = path)
```

mighty_repo_url	<i>URL component repo</i>
-----------------	---------------------------

Description

A component repo served as raw files under a base URL. Requires httr2 (>= 1.2.2).

See [mighty_repo\(\)](#) for the component layout. .mustache is tried before .R, and <name>/ before files directly under url. The first found is used. Status 404 and 410 count as not found; other errors are raised. name is percent-encoded, and names containing / are not found. Responses are not cached.

Listing components (see [list_components\(\)](#)) requires an HTML directory index at <url>/, and at <url>/<dir>/ for nested components, e.g. Apache or nginx autoindex, or python -m http.server. Listing aborts if an index is unavailable or not HTML.

Transient errors (see [httr2::req_retry\(\)](#)) and network failures are retried. The number of attempts is set by the max_tries option, and each attempt is limited by the timeout option (see [mighty.component-options](#)).

Usage

```
mighty_repo_url(url)
```

Arguments

url character(1) Base URL starting with http:// or https://.

Value

A mighty_repo_url object.

Examples

```
repo <- mighty_repo_url(
  url = paste0(
    "https://raw.githubusercontent.com/",
    "NovoNordisk-OpenSource/mighty.standards/main/components"
  )
)
format(repo)
```

```
find_component(component = "dummy", repos = repo)
```

mighty_repos	<i>Collection of component repos</i>
--------------	--------------------------------------

Description

An ordered collection of component repos. Lookups search the repos in order, and the first match wins.

Each element of repos is either a repo spec, passed to [mighty_repo\(\)](#), or an object inheriting from mighty_repo_class. Repos are created once when the collection is created.

Usage

```
mighty_repos(repos = character(0))
```

Arguments

repos character vector of repo specs, a single mighty_repo_class object, or a list of repo specs and mighty_repo_class objects.

Value

A mighty_repos object: a list of repos.

See Also

[find_component\(\)](#), [list_components\(\)](#)

Examples

```
path <- system.file("examples", package = "mighty.component")
repos <- mighty_repos(repos = c(path, paste0("local::", path)))
repos

find_component(component = "ady", repos = repos)
```

Index

`find_component`, [2](#)
`find_component()`, [4](#), [6](#), [19](#)

`get_component`, [3](#)
`get_component()`, [2](#), [6](#), [7](#), [10](#)
`get_rendered_component` (`get_component`),
[3](#)
`get_rendered_component()`, [5](#), [11](#), [12](#)
`get_test_component`, [4](#)
`get_test_component()`, [13](#), [15](#)
`gh::gh_token()`, [17](#)

`httr2::req_retry()`, [18](#)

`list_components`, [5](#)
`list_components()`, [2](#), [18](#), [19](#)

`mighty.component-options`, [6](#), [17](#), [18](#)
`mighty_component`, [2–4](#), [7](#), [11–13](#)
`mighty_component_rendered`, [3](#), [4](#), [10](#), [11](#),
[13](#)
`mighty_component_test`, [4](#), [5](#), [13](#)
`mighty_repo`, [16](#)
`mighty_repo()`, [2–6](#), [17–19](#)
`mighty_repo_class` (`mighty_repo`), [16](#)
`mighty_repo_github`, [17](#)
`mighty_repo_github()`, [7](#), [16](#)
`mighty_repo_local`, [18](#)
`mighty_repo_local()`, [16](#), [17](#)
`mighty_repo_url`, [18](#)
`mighty_repo_url()`, [7](#), [16](#)
`mighty_repos`, [19](#)
`mighty_repos()`, [2](#), [3](#), [5](#), [6](#), [16](#)

`whisker::whisker.render()`, [10](#)

`zephyr::verbosity_level`, [7](#)