Package: papillon (via r-universe)

September 19, 2024

Title Build And Highlight Package Documentation With Customized Templates

Version 0.0.1.9000

Description Helps creating company visual identity. Build your templates for your package vignettes, reports with {bookdown}, package presentation with {pkgdown}. Highlight your documentation in your projects. Allow for code folding.

License GPL-3

URL https://github.com/Thinkr-open/papillon

BugReports https://github.com/Thinkr-open/papillon/issues

Imports attachment, bookdown, cli, crayon, desc, magrittr, pkgdown, readr, rmarkdown, usethis, utils

Suggests devtools, knitr, remotes, shiny, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/Needs/website ThinkR-open/thinkrtemplate

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.2

Repository https://thinkr-open.r-universe.dev

RemoteUrl https://github.com/ThinkR-open/papillon

RemoteRef HEAD

RemoteSha e5183d023621fc3fa1b6cbbacb21a73905306ed9

Contents

`build_book` *Build book from vignettes*

Description

Build book from vignettes

Usage

```
build_book(
  path = "inst/report",
  path.v = "vignettes",
  output_format = c("bookdown::gitbook", "bookdown::pdf_document2"),
  clean_before = TRUE,
  clean_after = TRUE,
  keep_rmd = "index\\.Rmd$|zzz-references\\.Rmd$",
  clean = TRUE
)
```

Arguments

`path`	Path of the book
`path.v`	Path to vignettes folder to copy in the book folder
`output_format`	Output format of the book. "bookdown::gitbook", "bookdown::pdf_document2"
`clean_before`	Logical. Whether to remove all Rmd (except keep_rmd) before build
`clean_after`	Logical. Whether to remove all Rmd (except keep_rmd) after build
`keep_rmd`	Regex to list Rmd filenames to keep if clean_before or clean_after is TRUE. You'd better keep index.Rmd.
`clean`	Whether to delete the possible output files. If `FALSE`, simply print out a list of files/directories that should probably be deleted. You can set the global option `bookdown.clean_book = TRUE` to force this function to delete files. You are recommended to take a look at the list of files at least once before actually deleting them, i.e. run `clean_book(FALSE)` before `clean_book(TRUE)`.

build_pkgdown — *Build pkgdown site and move to inst*

Description

If "docs" is not in "inst" folder, it will not be available to the users

Usage

```
build_pkgdown(
  move = TRUE,
  clean_before = TRUE,
  clean_after = TRUE,
  yml,
  favicon,
  preview = NA,
  ...
)
```

Arguments

`move`	Logical. Whether to move the "docs" folder in "inst" to be kept in the package
`clean_before`	Logical. Whether to empty the "docs" and "inst/docs" prior to build site
`clean_after`	Logical. Whether to remove the original "docs" folder at the root of the project
`yml`	path to custom "_pkgdown.yml" file
`favicon`	path to favicon
`preview`	If `TRUE`, or `is.na(preview) && interactive()`, will preview freshly generated section in browser.
`...`	Other parameters needed by `build_site`

create_biblio_file — *Create bibliography md file from a vector of packages*

Description

Create bibliography md file from a vector of packages

Usage

```
create_biblio_file(
  packages,
  out.dir,
  output = c("packages", "references"),
  to = c("html", "markdown"),
  custom.md,
  edit = TRUE
)
```

Arguments

packages	vector of packages names
out.dir	Directory where to save output md file
output	Vector with "packages" (list of packages as bullet points), "references" (list of citation references) or both "packages" and "references"
to	Format to convert to. "html", "markdown" or "raw" text
custom.md	Vector of markdown text to add to the document before rendering
edit	Logical. Whether to open output md file for manual edition

Examples

```
packages <- c("rmarkdown", "attachment")
create_biblio_file(packages, out.dir = tempdir())
```

---

create_book *Create a directory for a book*

Description

Create a directory for a book

Usage

```
create_book(path = "inst/report", clean = FALSE, template)
```

Arguments

path	Book directory
clean	Logical. Whether to remove Chapter Rmd files.
template	A folder with personal template files for a bookdown site

Examples

```
## Not run:
book_folder <- tempdir()
create_book(path = book_folder, clean = TRUE)

## End(Not run)
```

`create_pkg_biblio_file` *Create bibliography md file from a package DESCRIPTION file*

Description

Create bibliography md file from a package DESCRIPTION file

Usage

```
create_pkg_biblio_file(
  path = "DESCRIPTION",
  out.dir,
  output = c("packages", "references"),
  to = c("html", "markdown"),
  custom.md,
  edit = TRUE
)
```

Arguments

`path`	Path to DESCRIPTION file
`out.dir`	Directory where to save output md file
`output`	Vector with "packages" (list of packages as bullet points), "references" (list of citation references) or both "packages" and "references"
`to`	Format to convert to. "html", "markdown" or "raw" text
`custom.md`	Vector of markdown text to add to the document before rendering
`edit`	Logical. Whether to open output md file for manual edition

create_pkg_desc_file *Create package description file*

Description

Create package description file

Usage

```
create_pkg_desc_file(
  path = "DESCRIPTION",
  source = c("archive", "github", "git"),
  url,
  out.dir,
  to = c("html", "markdown", "raw"),
  edit = TRUE
)
```

Arguments

path	Path to description file
source	Will the package be delivered as archive, from github or from another git?
url	url of the repository if not in URL in DESCRIPTION. Used when source is github or git.
out.dir	Path to save output file
to	Format to convert to. "html", "markdown" or "raw" text
edit	Logical. Whether to open file for edition

generate_readme_rmd *Create and update a custom README.Rmd with installation instructions*

Description

Create and update a custom README.Rmd with installation instructions

Usage

```
generate_readme_rmd(
  base_path = usethis::proj_get(),
  parts = c("description", "installation"),
  source = c("archive", "github", "git"),
  edit = TRUE,
  url
)
```

Arguments

base_path	Path of the package to generate Readme from
parts	Parts to modify in the Readme: c("description", "installation")
source	Will the package be delivered as archive, from github or from another git?
edit	Logical. Whether to open README.Rmd
url	url of the repository if not in URL in DESCRIPTION. Used when source is github or git.

Value

A README.Rmd file

Examples

```
## Not run:
generate_readme_rmd()

## End(Not run)
```

---

open_guide_function *Create a function inside R folder in your package to open userguide*

Description

Create a function inside R folder in your package to open userguide

Usage

```
open_guide_function(path = "inst/report")
```

Arguments

path	Path to the guide inside your package

open_pkgdown_function *Create a function inside R folder in your package to open pkgdown website*

Description

Create a function inside R folder in your package to open pkgdown website

Usage

```
open_pkgdown_function(path = "inst/docs")
```

Arguments

path	Path to the pkgdown inside your package

out_from_lines *write output in utf8*

Description

write output in utf8

Usage

```
out_from_lines(
  out_lines,
  to = c("html", "markdown", "raw"),
  out.dir,
  filename = "file",
  edit = TRUE
)
```

Arguments

out_lines	vector of html lines to write
to	Format to convert to. "html", "markdown" or "raw" text
out.dir	Directory where to save output md file
filename	filename without extentions
edit	Logical. Whether to open file for edition

`render_external` *Render Rmd in a script external to the current session*

Description

Render Rmd in a script external to the current session

Usage

```
render_external(
  input,
  output_format = "rmarkdown::html_document",
  output_file,
  output_options,
  warn = -1
)
```

Arguments

`input`	The input file to be rendered. This can be an R script (.R), an R Markdown document (.Rmd), or a plain markdown document.
`output_format`	The R Markdown output format to convert to. The option `"all"` will render all formats defined within the file. The option can be the name of a format (e.g. `"html_document"`) and that will render the document to that single format. One can also use a vector of format names to render to multiple formats. Alternatively, you can pass an output format object (e.g. `html_document()`). If using `NULL` then the output format is the first one defined in the YAML frontmatter in the input file (this defaults to HTML if no format is specified there). If you pass an output format object to `output_format`, the options specified in the YAML header or `_output.yml` will be ignored and you must explicitly set all the options you want when you construct the object. If you pass a string, the output format will use the output parameters in the YAML header or `_output.yml`.
`output_file`	The name of the output file. If using `NULL` then the output filename will be based on filename for the input file. If a filename is provided, a path to the output file can also be provided. Note that the `output_dir` option allows for specifying the output file path as well, however, if also specifying the path, the directory must exist. If `output_file` is specified but does not have a file extension, an extension will be automatically added according to the output format. To avoid the automatic file extension, put the `output_file` value in `I()`, e.g., `I('my-output')`.
`output_options`	List of output options that can override the options specified in metadata (e.g. could be used to force `self_contained` or `mathjax = "local"`). Note that this is only valid when the output format is read from metadata (i.e. not a custom format object passed to `output_format`).
`warn`	sets the handling of warning messages. -1 or 0

Index