Package 'roxytypes'

September 20, 2024

Title Typed Parameter Tags for Integration with 'roxygen2'

Version 0.1.1

Description Provides typed parameter documentation tags for integration with 'roxygen2'. Typed parameter tags provide a consistent interface for annotating expected types for parameters and returned values. Tools for converting from existing styles are also provided to easily adapt projects which implement typed documentation by convention rather than tag. Use the default format or provide your own.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/openpharma/roxytypes, https://openpharma.github.io/roxytypes/

BugReports https://github.com/openpharma/roxytypes/issues

Imports utils, cli, glue, roxygen2

Suggests mockery, testthat

Enhances roxylint

Config/Needs/documentation roxylint

RoxygenNote 7.3.1

NeedsCompilation no

Author Doug Kelkhoff [aut, cre]

Maintainer Doug Kelkhoff <doug.kelkhoff@gmail.com>

Repository CRAN

Date/Publication 2024-09-20 12:10:02 UTC

Contents

`build_format_regex` *Build format regular expression*

Description

Allow `glue`-style formatting using keyworded regular expressions. The original `glue` string (anything that isn't expanded by `glue`) is treated as a string literal, whereas the contents of populated values can be regular expressions, allowing for a more user-friendly way to construct complicated regular expressions.

Usage

```
build_format_regex(
  format,
  format_re,
  ...,
  type = re_backticked(),
  description = re_any()
)

re_backticked()

re_any()

escape_non_glue_re(x)
```

Arguments

`format`	(`character[1]`) A `glue`-style format string. Expanded whisker values are used as a shorthand for capture groups, where ellipsis arguments can be provided for additional capture group patterns.
`format_re`	(`character[1]`) Alternatively, provide a standard regular expression directly.
`...`	Additional arguments provide keyworded capture groups for `format`
`type`	(`character[1]`) A regular expression to use to match a type signature. By default, matches within backticks.

description	(character[1]) A regular expression to use to match a parameter description. By default, matches any string.
x	(character[1]) A string to escape.

Details

To bypass glue entirely and use a standard regular expression, use format_re.

The provided regular expression must match all characters from the start of a string to the end. The string also matches using "dot all" syntax, meaning that the . expression will also match newline characters.

Value

(character[1]:) A regular expression string, built from component sub-expressions.

Functions

- re_backticked(): Match within backticks
- re_any(): Match any
- escape_non_glue_re(): Escape all regular expression special characters
  In addition, avoid escaping {}'s that appear to be used as glue keywords. Handles only simple cases, and does not handle recusive curly nesting.

Examples

```
re <- roxytypes:::build_format_regex(
  "{as}{any}{bs}",
  as = "a+",
  bs = "b+",
  any = ".*?"
)

roxytypes:::regex_capture(re, "aaaa\n\nbb", perl = TRUE)

text <- "@param (`test(\")\")`)"

pattern <- sprintf("`%s`", re_backticked())

m <- regexec(pattern, text, perl = TRUE)
regmatches(text, m)[[1]]
# [1] "`test(\")\")`"

# curlies escaped, as this does not appear to be a glue-style usage
roxytypes:::escape_non_glue_re(".{1,3}")

# curlies not escaped, as this is a glue-style usage
roxytypes:::escape_non_glue_re("this is a {test}")
```

convert *Convert* roxygen2 *tags to* roxytypes *tags*

Description

Convert a package codebase into applicable roxytypes tags. For roxygen2 tags with drop-in replacements (namely @param and @return tags), process descriptions and replace tags with roxytypes equivalents.

Usage

```
convert(
  path = ".",
  format = config(path, refresh = TRUE, cache = FALSE)$format,
  ...,
  unmatched = FALSE,
  verbose = interactive()
)
```

Arguments

path	(character[1]) A file path within your package. Defaults to the current working directory.
format	(character[1]) A glue-style format to use to parse types and descriptions for conversion to roxytypes tags. Available glue keywords include type and description. By default, type will match any string until a closing backtick and description will match any string. See details for more information.
...	Additional arguments passed to build_format_regex().
unmatched	(logical[1]) Indicates whether tags that fail to match should still be converted into roxytypes tags. Such conversions may be convenient if you aim to convert your package holistically, as it will help to flag undocumented parameter types the next time you re-build your documentation.
verbose	(logical[1]) Indicates whether command-line interface should be emitted so that changes can be reviewed interactively.

Details

A format string is built using build_format_regex(), which accepts parameters type and description, which describe how to match these components of a parameter definition. They are combined with the literal content of format to produce a regular expression to split existing definitions.

For comprehensive control, pass format_re directly, bypassing expression construction altogether.

Value

(logical[1]) TRUE if successfully completes, FALSE if aborted. Always returns invisibly.

Examples

```
## Not run:
convert("(`{type}`) {description}")

## End(Not run)
```

---

default_format *Default formatter for* `@typed`

Description

Adds special cases for when the type uses other roxygen2 syntax

Usage

```
default_format(x, name, type, description, ...)
```

Arguments

`x`	(`roxygen2::roxy_tag()`) The tag to format.
`name`, `type`, `description`	(`character(1)`) Fields parsed from the `@typed` tag.
`...`	Additional arguments unused.

Value

A formatted character value.

---

read_dcf_asis *A helper to reliably read DCF files*

Description

A helper to reliably read DCF files

Usage

```
read_dcf_asis(path)
```

Arguments

`path`	(`character[1]`) A file path to a DESCRIPTION file.

Value

(`data.frame`) The result of `read.dcf()`.

roxy_tag_parse.roxy_tag_typed

roxygen2 `@typed` *tag parsing*

Description

Parse a `@typed` tag and return parsed components as value

Usage

```
## S3 method for class 'roxy_tag_typed'
roxy_tag_parse(x)
```

Arguments

x	A tag

Value

(`roxygen2 tag`) A parsed `roxygen2` `@typed` rd_tag.

roxy_tag_parse.roxy_tag_typedreturn

roxygen2 `@typedreturn` *tag parsing*

Description

Parse a `@typedreturn` tag and return parsed components as value

Usage

```
## S3 method for class 'roxy_tag_typedreturn'
roxy_tag_parse(x)
```

Arguments

x	A tag

Value

(`roxygen2 tag`) A parsed `roxygen2` `@typedreturn` rd_tag.

roxy_tag_rd.roxy_tag_typed

roxygen2 @typed *tag rd section population*

Description

Push typed fields into @param section

Usage

```
## S3 method for class 'roxy_tag_typed'
roxy_tag_rd(x, base_path, env)
```

Arguments

x	The tag
base_path	Path to package root directory.
env	Environment in which to evaluate code (if needed)

Value

(roxygen2::rd_section) A roxygen2 "param" rd_section with formatted type information.

roxy_tag_rd.roxy_tag_typedreturn

roxygen2 @typedreturn *tag rd section population*

Description

Push typed fields into @param section

Usage

```
## S3 method for class 'roxy_tag_typedreturn'
roxy_tag_rd(x, base_path, env)
```

Arguments

x	The tag
base_path	Path to package root directory.
env	Environment in which to evaluate code (if needed)

Value

(roxygen2::rd_section) A roxygen2 @value rd_tag with formatted type information.

tags roxytypes *tags*

Description

The `@typed` tag introduces a syntax for defining parameter types as a `roxygen2` tag.

Usage

```
#' @typed <var>: <type>
#'   <description>
```

Details

Be aware that there are a few syntactic requirements:

- `:` delimiter between the variable name and type.
- `\n` after the type to separate it from the description.

Default `type` Parsing Syntax

The type portion of the `@typed` tag syntax will handle a bit of syntax as special cases.

- `[type]`: Types wrapped in brackets, for example `[roxygen2::roxy_tags()]` will be left as-is, without wrapping the string in backticks to display as inline code and preserve the native `roxygen2` reference link.

  ```
  #' @typed arg: [package::function()]
  #'   long form description.
  ```

- `` `type` ``: Types wrapped in backticks will be kept as-is. Additional backticks will not be inserted.

  ```
  #' @typed arg: `class`
  #'   long form description.
  ```

- `"type"` or `'type'`: Types wrapped in quotes (either single or double), will be provided as literal values, removing the surrounding quotation marks.

  ```
  #' @typed arg: "`class_a` or `class_b`"
  #'   depending on the class of the object provided, either an `"A"`
  #'   or a `"B"`.
  ```

Custom `type` Parsing Function

The above defaults are meant to cover most use cases and should be sufficient for all but the most elaborate development practices. If you need to go beyond these default behaviors, you can also provide a parsing function, accepting the parsed roxygen tag as well as the raw contents.

The function accepts the roxygen2::roxy_tag() produced when parsing the tag, whose $val contains fields name, type and description. For convenience, the $val contents is unpacked as arguments, though the structure of this tag is liable to change.

To implement a typescript-style class union syntax,

```
#' @typed arg: class_a | class_b | class_c
#'   depending on the class of the object provided, either an `"A"`
#'   or a `"B"`.
```

to produce the parameter definition

```
(`class_a`, `class_c` or `class_b`) depending on the class of the object
provided, either an `"A"`, `"B"` or a `"C"`.
```

we might define the following in DESCRIPTION (or in man/roxytypes/meta.R).

```
Config/roxytypes: list(
  format = function(tag, ..., name, type, description) {
    types <- paste0("`", trimws(strsplit(type, "|", fixed = TRUE)[[1]]), "`")
    types <- glue::glue_collapse(types, sep = ", ", last = " or ")
    paste0("(", types, ") ", description)
  }
)
```

---

typedreturn roxygen2 @typedreturn *tag*

Description

The @typedreturn tag introduces a syntax for defining return types as a roxygen2 tag.

Usage

```
#' @typedreturn <type>
#'   <description>
```

Details

There is one important syntactic features:

- \n after the type to separate it from the description.

`with_roxy_field_subclass` *A helper to apply field names to all roxy_tag val fields*

Description

A helper to apply field names to all roxy_tag val fields

Usage

```
with_roxy_field_subclass(x)
```

Arguments

`x`	(`list`) A named list of tag val contents

Value

(`: list`) A nearly identical list, where elements have additional subclasses based on their field names.

Index