Package ‘roxygen2’

January 29, 2014

Title In-source documentation for R

Description A Doxygen-like in-source documentation system for Rd, collation, and NAMESPACE.

Version 3.1.0

License GPL (>= 2)

Depends R (>= 3.0.2)

Imports stringr (>= 0.5), tools, brew, digest, codetools, methods,Rcpp

Suggests testthat

LinkingTo Rcpp (>= 0.10.6)

Roxygen list(wrap = FALSE)

Collate 'RcppExports.R' 'alias.R' 'description.R' 'family.R'
'inherit-params.R' 'object-defaults.R' 'object-from-call.R'
'object.R' 'parse-preref.R' 'parse-registry.R' 'parse.R' 'rc.R'
'rd-escape.R' 'rd-file-api.R' 'rd-parse.R' 'rd-tag-api.R'
'roclet-collate.R' 'roclet-namespace.R' 'roclet-rd.R'
'roclet.R' 'roxygen.R' 'roxygenize.R' 's3.R' 'source.R'
'template.R' 'topic-name.R' 'topo-sort.R' 'usage.R' 'util-locale.R' 'utils.R'

Author Hadley Wickham [aut, cre],Peter Danenberg [aut],Manuel Eugster [aut]

Maintainer Hadley Wickham <`h.wickham@gmail.com`>

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-01-29 16:24:19

R topics documented:

`is_s3_generic` *Determine if a function is an S3 generic or S3 method.*

Description

`is_s3_generic` compares name to `.knownS3Generics` and `.S3PrimitiveGenerics`, then uses `findGlobals` to see if the functionion calls `UseMethod`.

`is_s3_method` builds names of all possible generics for that function and then checks if any of them actually is a generic.

Usage

```
is_s3_generic(name, env = parent.frame())
```

Arguments

`name`	name of function.
`env`	environment to search in.

`namespace_roclet` *Roclet: make NAMESPACE.*

Description

This roclet automates the production of a ‘`NAMESPACE`’ file, see *Writing R Extensions* (`http://cran.r-project.org/doc/manuals/R-exts.pdf`) for details.

Usage

```
namespace_roclet()
```

Tags

There are four tags for exporting objects from the package:

`@export` Roxygen guesses the directive: `export` for functions, `exportMethod` for S4 methods, `S3method` for S3 methods, `exportClass` for S4 classes.

This is the only directive you should need for documented function, the other directives are useful if you want to export (e.g.) methods but not document them.

`@export f g ...` overrides auto-detection and produces multiple export directives: `export(f)`, `export(g)` ...

`@exportClass x` produces `exportClasses(x)` directive.

`@exportMethod x` produces `exportMethods(x)` directive.

`@S3method generic class` produces `S3method(generic,class)` directive

There are five tags for importing objects into the package:

`@import package` produces `import(package) directive to import all functions  from the given package`

`@importFrom package functiona functionb ...` produces multiple `importFrom(package, function)` directives to import selected functions from a package.

`@importClassesFrom package classa classb ...` produces multiple `importClassesFrom(package, class)` directives to import selected classes from a package.

`@importMethodsFrom package methoda methodb ...` produces multiple `importMethodsFrom(package, method)` directives to import selected methods from a package.

`@useDynLib package` produces a `useDynLib(package)` directive to import all compiled routines from the shared objects in the specified package

`@useDynLib paackage routinea routineb` produces multiple `useDynLib(package,routine)` directions to import specified compiled routines from a package.

Only unique directives are saved to the ‘`NAMESPACE`’ file, so you can repeat them as needed to maintain a close link between the functions where they are needed and the namespace file.

See Also

Other roclets: `rd_roclet`

Examples

```
#' An example file, example.R, which imports
#' packages foo and bar
#' @import foo bar
NULL

#' An exportable function
#' @export
fun <- function() {}

roclet <- namespace_roclet()
## Not run: roc_proc(roclet, "example.R")
## Not run: roc_out(roclet, "example.R", ".")
```

`rd_roclet` *Roclet: make Rd files.*

Description

This roclet is the workhorse of **roxygen**, producing the Rd files that document that functions in your package.

Usage

```
rd_roclet()
```

Details

This roclet also automatically runs `checkRd` on all generated Rd files so that you know as early as possible if there's a problem.

Required tags

As well as a title and description, extracted from the first sentence and first paragraph respectively, all functions must have the following tags:

`@param name description` Document a parameter. Documentation is required for every parameter.

`@inheritParams source_function` Alternatively, you can inherit parameter description from another function. This tag will bring in all documentation for parameters that are undocumented in the current function, but documented in the source function. The source can be a function in the current package, `function`, or another package `package::function`.

`@method generic class` Required if your function is an S3 method. This helps R to distinguish between (e.g.) `t.test` and the `t` method for the `test` class.

Optional tags that add extra information

Valid tags for `rd_roclet` are:

`@examples R code...` Highly recommended: example code that demonstrates how to use your function. Use \dontrun to tag code that should not automatically be run.

`@example path/relative/to/packge/root` Instead of including examples directly in the documentation, you can include them as separate files, and use the `@example` tag to insert them into the documentation.

`@return` Used to document the object returned by the function. For lists, use the `\item{name a}{description a}` describe each component of the list

`@author authors...` A free text string describing the authors of the function. This is typically only necessary if the author is not the same as the package author.

`@note contents` Create a note section containing additional information.

`@section Name: contents` Use to add to an arbitrary section to the documentation. The name of the section will be the content before the first colon, and the contents will be everything after the colon.

`@keywords keyword1 keyword2 ...` Keywords are optional, but if present, must be taken from the list in ‘`file.path(R.home(), "doc/KEYWORDS")`’. Use the `internal` keyword for functions that should not appear in the main function listing.

Optional tags for cross-referencing

`@aliases space separated aliases` Add additional aliases, through which the user can find the documentation with ?. The topic name is always included in the list of aliases.

`@concepts space separated concepts` Similar to `@aliases` but for `help.search`

`@references free text reference` Pointers to the literature related to this object.

`@seealso Text with \code{\link{function}}` Pointers to related R objects, and why you might be interested in them.

`@family family name` Automatically adds see-also cross-references between all functions in a family. A function can belong to multiple families.

Template tags

Templates make it possible to substantially reduce documentation duplication. A template is an ‘`R`’ file in the `man-roxygen/` directory. It is processed with `brew` and then inserted into the roxygen block. You can run any R code with brew; you can insert template variables with `<%= varname %>`.

`@template templateName` Insert named template in current location.

`@templateVar varname value` Set up variables for template use.

Limitations:

- Templates are not parsed recursively, so you can not include templates from within other templates.
- Templates must be composed of complete tags - becuase all roxygen tags are current block tags, they can not be used for inline insertions.

Optional tags that override defaults

These tags all override the default values that roxygen guess from inspecting the source code.

`@rdname filename` Overrides the output file name (without extension). This is useful if your function has a name that is not a valid filename (e.g. `[[<-`), or you want to merge documentation for multiple function into a single file.

`@title Topic title` Specify the topic title, which by by default is taken from the first sentence of the roxygen block.

`@usage usage_string` Override the default usage string. You should not need to use this tag - if you are trying to document multiple functions in the same topic, use `@rdname`.

Tags for non-functions

These tags are useful when documenting things that aren't functions, datasets and packages.

`@name topicname` Override the default topic name, which is taken by default from the object that is assigned to in the code immediately following the roxygen block. This tag is useful when documenting datasets, and other non-function elements.

`@docType type` Type of object being documented. Useful values are `data` and `package`. Package doc type will automatically add a `package-` alias if needed.

`@format description` A textual description of the format of the object.

`@source text` The original source of the data.

`@slot name description` Describe the slots of an S4 class in a standard way. Slots will be listed in their own section.

See Also

Other roclets: `S3method`, `export`, `exportClass`, `exportMethod`, `import`, `importClassesFrom`, `importFrom`, `importMethodsFrom`, `namespace_roclet`

Examples

```
roclet <- rd_roclet()
## Not run: roc_proc(roclet, "example.R")
## Not run: roc_out(roclet, "example.R", ".")
```

---

roxygen *In-line documentation for R.*

Description

Roxygen is a Doxygen-like documentation system for R; allowing in-source specification of Rd files, collation and namespace directives.

Details

If you have existing Rd files, check out the `Rd2roxygen` package for a convenient way of converting Rd files to roxygen comments.

Author(s)

Hadley Wickham <`h.wickham@gmail.com`>, Peter Danenberg <`pcd@roxygen.org`>, Manuel Eugster <`Manuel.Eugster@stat.uni-muenchen.de`>

Maintainer: Hadley Wickham <`h.wickham@gmail.com`>

See Also

See `namespace_roclet`, `rd_roclet`, for an overview of roxygen tags.

Examples

```
## Not run: roxygenize('pkg')
```

`roxygenize` *Process a package with the Rd, namespace and collate roclets.*

Description

This is the workhorse function that uses roclets, the built-in document tranformation functions, to build all documentation for a package. See the documentation for the individual roclets, `rd_roclet`, `namespace_roclet`, and for `update_collate`, for more details.

Usage

```
roxygenize(package.dir = ".", roxygen.dir = package.dir,
  copy.package = package.dir != roxygen.dir, overwrite = TRUE,
  unlink.target = FALSE, roclets = NULL, load_code = source_package)

roxygenise(package.dir = ".", roxygen.dir = package.dir,
  copy.package = package.dir != roxygen.dir, overwrite = TRUE,
  unlink.target = FALSE, roclets = NULL, load_code = source_package)
```

Arguments

`package.dir`	the package's top directory
`roxygen.dir,copy.package,overwrite,unlink.target`	deprecated
`roclets`	character vector of roclet names to apply to package. This defaults to `NULL`, which will use the `roclets` fields in the list provided in the `Roxygen` DESCRIPTION field. If none are specified, defaults to `c("collate", "namespace", "rd")`.
`load_code`	A function used to load all the R code in the package directory. It is called with the path to the package, and it should return an environment containing all the sourced code.

Value

`NULL`

update_collate — *Update Collate field in DESCRIPTION.*

Description

Topologically sort R files and record in Collate field. The topological sort is based on the `@include` tag, which should specify the filenames (space separated) that should be loaded before the current file - these are typically necessary if you're using S4 or RC classes (because super classes must be defined before subclasses). If there are no `@include` tags Collate will be left blank, indicating that the order of loading does not matter.

Usage

```
update_collate(base_path)
```

Arguments

`base_path`	Path to package directory

Details

This is not a roclet because roclets need the values of objects in a package, and those values can not be generated unless you've sourced the files, and you can't source the files unless you know the correct order.

Examples

```
#' `example-a.R', `example-b.R' and `example-c.R' reside
#' in the `example' directory, with dependencies
#' a -> {b, c}. This is `example-a.R'.
#' @include example-b.R
#' @include example-c.R
NULL

## Not run:
  update_collate("my_package")

## End(Not run)
```

Index