

Package ‘R.methodsS3’

June 14, 2022

Version 1.8.2

Depends R (>= 2.13.0)

Imports utils

Suggests codetools

Title S3 Methods Simplified

Author Henrik Bengtsson [aut, cre, cph]

Maintainer Henrik Bengtsson <henrikb@braju.com>

Description Methods that simplify the setup of S3 generic functions and S3 methods. Major effort has been made in making definition of methods as simple as possible with a minimum of maintenance for package developers. For example, generic functions are created automatically, if missing, and naming conflict are automatically solved, if possible. The method `setMethodS3()` is a good start for those who in the future may want to migrate to S4. This is a cross-platform package implemented in pure R that generates standard S3 methods.

License LGPL (>= 2.1)

LazyLoad TRUE

URL <https://github.com/HenrikBengtsson/R.methodsS3>

BugReports <https://github.com/HenrikBengtsson/R.methodsS3/issues>

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-13 22:00:14 UTC

R topics documented:

R.methodsS3-package	2
getGenericS3	3
getMethodS3	4
isGenericS3	4
setGenericS3	5
setMethodS3	6
Index	9

R.methodsS3-package *Package R.methodsS3*

Description

Methods that simplify the setup of S3 generic functions and S3 methods. Major effort has been made in making definition of methods as simple as possible with a minimum of maintenance for package developers. For example, generic functions are created automatically, if missing, and naming conflict are automatically solved, if possible. The method `setMethodS3()` is a good start for those who in the future may want to migrate to S4. This is a cross-platform package implemented in pure R that generates standard S3 methods. This contents of this package originates from the **R.oo** package [1].

Installation and updates

To install this package do

```
install.packages("R.methodsS3")
```

To get the "devel" version, see <https://github.com/HenrikBengtsson/R.methodsS3/>.

Dependencies and other requirements

This package only requires a standard R installation.

To get started

To get started, see:

1. `setMethodS3()` - Simple and safe creation of S3 methods and, whenever needed, automatic creation of S3 generic function.

Further readings

For a detailed introduction to the package, see [1].

How to cite this package

Whenever using this package, please cite [1] as

Bengtsson, H. The R.oo package - Object-Oriented Programming with References Using Standard R Code, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), ISSN 1609-395X, Hornik, K.; Leisch, F. & Zeileis, A. (ed.), 2003

License

The releases of this package is licensed under LGPL version 2.1 or newer.

Author(s)

Henrik Bengtsson

References

[1] H. Bengtsson, *The R.oo package - Object-Oriented Programming with References Using Standard R Code*, In Kurt Hornik, Friedrich Leisch and Achim Zeileis, editors, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20-22, Vienna, Austria. <https://www.r-project.org/conferences/DSC-2003/Proceedings/>

getGenericS3	<i>Gets an S3 generic function</i>
--------------	------------------------------------

Description

Gets an S3 generic function.

Usage

```
## Default S3 method:  
getGenericS3(name, envir=parent.frame(), inherits=TRUE, ...)
```

Arguments

name	The name of the generic function.
envir	The environment from which the search for the generic function is done.
inherits	A logical specifying whether the enclosing frames should be searched or not.
...	Not used.

Author(s)

Henrik Bengtsson

See Also

[setGenericS3\(\)](#). [getMethodS3\(\)](#). [isGenericS3\(\)](#).

getMethodS3	<i>Gets an S3 method</i>
-------------	--------------------------

Description

Gets an S3 method.

Usage

```
## Default S3 method:
getMethodS3(name, class="default", envir=parent.frame(), ...)
```

Arguments

name	The name of the method.
class	The class of the method.
envir	The environment from which the search for the S3 method is done.
...	Not used.

Author(s)

Henrik Bengtsson

See Also

This is just a conveniency wrapper around [getS3method](#) that have arguments consistent with [setMethodS3\(\)](#). [getGenericS3\(\)](#).

isGenericS3	<i>Checks if a function is a S3 generic function</i>
-------------	--

Description

Checks if a function is a S3 generic function.

Usage

```
## Default S3 method:
isGenericS3(fcn, envir=parent.frame(), ...)
```

Arguments

fcn	A function or a character string.
envir	If argument fcn is a character , this is the environment from which the search for the function is done.
...	Not used.

Details

A function is considered to be a generic S3/UseMethod function if its name matches one of the known S3 generic functions, or if it calls UseMethod().

Value

Returns `TRUE` if a generic S3/UseMethod function, otherwise `FALSE`.

Author(s)

Henrik Bengtsson

setGenericS3	<i>Creates an S3 generic function</i>
--------------	---------------------------------------

Description

Note that this method is a internal method called by `setMethodS3()` and there is no reason for calling it directly!

Creates a generic function in S3 style, i.e. setting a function with name `name` that dispatches the method `name` via UseMethod. If there is already a function named `name` that function is renamed to `name.default`.

Usage

```
## Default S3 method:
setGenericS3(name, export=TRUE, envir=parent.frame(), dontWarn=getOption("dontWarnPkgs"),
  validators=getOption("R.methodsS3:validators:setGenericS3"), overwrite=FALSE, ...)
```

Arguments

<code>name</code>	The name of the generic function.
<code>export</code>	A <code>logical</code> setting attribute "export".
<code>envir</code>	The environment for where this method should be stored.
<code>dontWarn</code>	If a non-generic method with the same name is found it will be "renamed" to a default method. If that method is found in a package with a name that is <i>not</i> found in <code>dontWarn</code> a warning will be produced, otherwise it will be renamed silently.
<code>validators</code>	An optional <code>list</code> of <code>functions</code> that can be used to assert that the generated generic function meets certain criteria.
<code>...</code>	Not used.
<code>overwrite</code>	If <code>TRUE</code> an already existing generic function with the same name will be overwritten, otherwise not.

Author(s)

Henrik Bengtsson

See Also

To define a method for a class see [setMethodS3\(\)](#). For more information about S3, see [UseMethod\(\)](#).

Examples

```
myCat.matrix <- function(..., sep=" ") {
  cat("A matrix:\n")
  cat(..., sep=sep)
  cat("\n")
}

myCat.default <- function(..., sep=" ") {
  cat(..., sep=sep)
  cat("\n")
}

setGenericS3("myCat")

myCat(1:10)
mat <- matrix(1:10, ncol=5)
myCat(mat)
```

 setMethodS3

Creates an S3 method

Description

Creates an S3 method. A function with name `<name>.<class>` will be set to definition. The method will get the modifiers specified by `modifiers`. If there exists no generic function for this method, it will be created automatically.

Usage

```
## Default S3 method:
setMethodS3(name, class="default", definition, private=FALSE, protected=FALSE,
  export=FALSE, static=FALSE, abstract=FALSE, trial=FALSE, deprecated=FALSE,
  envir=parent.frame(), overwrite=TRUE, conflict=c("warning", "error", "quiet"),
  createGeneric=TRUE, exportGeneric=TRUE, appendVarArgs=TRUE,
  validators=getOption("R.methodsS3:validators:setMethodS3"), ...)
```

Arguments

name	The name of the method.
class	The class for which the method should be defined. If <code>class == "default"</code> a function with name <code><name>.default</code> will be created.
definition	The method definition.
private, protected	If <code>private=TRUE</code> , the method is declared private. If <code>protected=TRUE</code> , the method is declared protected. In all other cases the method is declared public.
export	A logical setting attribute "export".
static	If TRUE this method is defined to be static, otherwise not. Currently this has no effect expect as an indicator.
abstract	If TRUE this method is defined to be abstract, otherwise not. Currently this has no effect expect as an indicator.
trial	If TRUE this method is defined to be a trial method, otherwise not. A trial method is a method that is introduced to be tried out and it might be modified, replaced or even removed in a future release. Some people prefer to call trial versions, beta version. Currently this has no effect expect as an indicator.
deprecated	If TRUE this method is defined to be deprecated, otherwise not. Currently this has no effect expect as an indicator.
envir	The environment for where this method should be stored.
overwrite	If TRUE an already existing generic function and an already existing method with the same name (and of the same class) will be overwritten, otherwise not.
conflict	If a method already exists with the same name (and of the same class), different actions can be taken. If "error", an exception will be thrown and the method will not be created. If "warning", a warning will be given and the method <i>will</i> be created, otherwise the conflict will be passed unnoticed.
createGeneric, exportGeneric	If <code>createGeneric=TRUE</code> , a generic <code>S3/UseMethod</code> function is defined for this method, iff missing, and <code>exportGeneric</code> species attribute "export" of it.
appendVarArgs	If TRUE , argument <code>...</code> is added with a warning, if missing. For special methods such as <code>\$</code> and <code>[[</code> , this is never done (argument is ignored). This will increase the chances that the method is consistent with a generic function with many arguments and/or argument <code>...</code>
validators	An optional list of functions that can be used to assert that the generated method meets certain criteria.
...	Passed to setGenericS3() , iff called.

Author(s)

Henrik Bengtsson

See AlsoFor more information about S3, see [UseMethod\(\)](#).

Examples

```
#####
# Example 1
#####
setMethodS3("foo", "default", function(x, ...) {
  cat("In default foo():\n");
  print(x, ...);
})

setMethodS3("foo", "character", function(s, ...) {
  cat("In foo() for class 'character':\n");
  print(s, ...);
})

# The generic function is automatically created!
print(foo)

foo(123)
foo("123")

#####
# Example 2
#
# Assume that in a loaded package there is already a function bar(),
# but you also want to use the name 'bar' for the character string.
# It may even be the case that you do not know of the other package,
# but your users do!
#####
# bar() in other package
bar <- function(x, y, ...) {
  cat("In bar() of 'other' package.\n");
}

# Your definition; will redefine bar() above to bar.default().
setMethodS3("bar", "character", function(object, ...) {
  cat("In bar() for class 'character':\n");
  print(object, ...);
})

bar(123)
bar("123")
```


Index

* **methods**

- getGenericS3, 3
- getMethodS3, 4
- isGenericS3, 4
- setGenericS3, 5
- setMethodS3, 6

* **package**

- R.methodsS3-package, 2

* **programming**

- getGenericS3, 3
- getMethodS3, 4
- isGenericS3, 4
- setGenericS3, 5
- setMethodS3, 6

character, 4

environment, 3, 4

FALSE, 5

function, 3–5, 7

getGenericS3, 3, 4

getMethodS3, 3, 4

getS3method, 4

isGenericS3, 3, 4

list, 5, 7

logical, 3, 5, 7

R.methodsS3 (R.methodsS3-package), 2

R.methodsS3-package, 2

setGenericS3, 3, 5, 7

setMethodS3, 2, 4–6, 6

TRUE, 5, 7

UseMethod, 6, 7

warning, 7