

Package ‘R.methodsS3’

January 2, 2012

Version 1.2.1

Date 2010-09-18

Title Utility function for defining S3 methods

Author Henrik Bengtsson <henrikb@braju.com>

Maintainer Henrik Bengtsson <henrikb@braju.com>

Depends R (>= 2.2.0)

Imports utils

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 want to migrate to S4. This is a cross-platform package implemented in pure R and generates standard S3 methods.

License LGPL (>= 2.1)

URL <http://www.braju.com/R/>

DevelURL <http://www.braju.com/R/>

LazyLoad TRUE

Repository CRAN

Repository/R-Forge/Project r-dots

Repository/R-Forge/Revision 409

Date/Publication 2010-09-25 06:18:00

R topics documented:

R.methodsS3-package	2
findDispatchMethodsS3	3
getDispatchMethodS3	4
getGenericS3	5
getMethodS3	5
setMethodS3	7
throw	9

Index	11
--------------	-----------

R.methodsS3-package	<i>Package R.methodsS3</i>
---------------------	----------------------------

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 want to migrate to S4. This is a cross-platform package implemented in pure R and generates standard S3 methods. This contents of this package was extracted from the **R.oo** package [1].

Installation and updates

To install this package do

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

To get the "devel" version, see <http://www.braju.com/R/>.

Dependancies 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

```
@INPROCEEDINGS{BengtssonH_2003,
  author      = {Henrik Bengtsson},
  title       = {The {R.oo} package - Object-Oriented Programming
                with References Using Standard {R} Code},
  booktitle   = {Proceedings of the 3rd International Workshop on
                Distributed Statistical Computing (DSC 2003)},
  year        = {2003},
  editor      = {Kurt Hornik and Friedrich Leisch and Achim Zeileis},
  address     = {Vienna, Austria},
  month       = {March},
  issn        = {1609-395X},
  howpublished = {http://www.ci.tuwien.ac.at/Conferences/DSC-2003/},
}
```

License

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

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

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. <http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/>

findDispatchMethodsS3 *Finds the S3 methods that a generic function would call*

Description

Finds the S3 methods that a generic function would call, ordered according to an S3 `class()` `vector`.

Usage

```
## Default S3 method:
findDispatchMethodsS3(methodName, classNames, firstOnly=FALSE, ...)
```

Arguments

methodName	A character string specifying the name of a generic function.
classNames	A character vector of <code>class()</code> names.
firstOnly	If <code>TRUE</code> , only the first method is returned.
...	Not used.

Value

Returns a names [list](#) structure.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[getDispatchMethodS3\(\)](#).

getDispatchMethodS3 *Gets the S3 method that a generic function would call*

Description

Gets the S3 method that a generic function would call according to an S3 [class\(\)](#) [vector](#).

Usage

```
## Default S3 method:
getDispatchMethodS3(methodName, classNames, ...)
```

Arguments

methodName	A character string specifying the name of a generic function.
classNames	A character vector of <code>class()</code> names.
...	Not used.

Value

Returns a [function](#), or throws an exception if not found.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

[findDispatchMethodsS3\(\)](#).

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

Description

Get an S3 generic function.

Usage

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

Arguments

name	The name of the generic function.
envir	The environment from which the search for the generic function is done.
...	Not used.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

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

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

Description

Get 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 (<http://www.braju.com/R/>)

See Also

This is just a conveniency wrapper around `getS3method` that have arguments consistent with `setMethodS3()`.
`getGenericS3()`.

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")
```

 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, static=FALSE, abstract=FALSE)
```

Arguments

<code>name</code>	The name of the method.
<code>class</code>	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.
<code>definition</code>	The method definition.
<code>private, protected</code>	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.
<code>static</code>	If <code>TRUE</code> this method is defined to be static, otherwise not. Currently this has no effect except as an indicator.
<code>abstract</code>	If <code>TRUE</code> this method is defined to be abstract, otherwise not. Currently this has no effect except as an indicator.
<code>trial</code>	If <code>TRUE</code> 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 except as an indicator.
<code>deprecated</code>	If <code>TRUE</code> this method is defined to be deprecated, otherwise not. Currently this has no effect except as an indicator.
<code>envir</code>	The environment for where this method should be stored.
<code>overwrite</code>	If <code>TRUE</code> an already existing method with the same name (and of the same class) will be overwritten, otherwise not.
<code>conflict</code>	If a method already exists with the same name (and of the same class), different actions can be taken. If <code>"error"</code> , an exception will be thrown and the method will not be created. If <code>"warning"</code> , a <code>warning</code> will be given and the method <i>will</i> be created, otherwise the conflict will be passed unnoticed.

createGeneric	If TRUE , a generic S3/UseMethod function is defined for this method.
appendVarArgs	If TRUE , argument ... is added with a warning, if missing. For special methods such as \$ and [], this is never done. This will increase the chances that the method is consistent with a generic function with many arguments and/or argument ...
validators	An optional list of functions that can be used to assert that the generated method meets certain criteria.
...	Not used.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

See Also

For 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")
```

throw	<i>Throws an exception</i>
-------	----------------------------

Description

Throws an exception by calling `stop()`.

Note that `throw()` can be defined for specific classes, which can then be caught (or not) using [tryCatch\(\)](#).

*This default function will be overridden by ditto in the **R.oo** package, if that is loaded. The latter [throw](#) implementation is fully backward compatible with this one, but the error object thrown is of class [Exception](#).*

Usage

```
## Default S3 method:
throw(...)
```

Arguments

... One or several strings that are concatenated and collapsed into one message string.

Value

Returns nothing.

Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

Examples

```
rbern <- function(n=1, prob=1/2) {  
  if (prob < 0 || prob > 1)  
    throw("Argument 'prob' is out of range: ", prob)  
  rbinom(n=n, size=1, prob=prob)  
}  
  
rbern(10, 0.4)  
# [1] 0 1 0 0 0 1 0 0 1 0  
tryCatch(rbern(10, 10*0.4),  
  error=function(ex) {}  
)
```

Index

- *Topic **error**
 - throw, 9
- *Topic **methods**
 - findDispatchMethodsS3, 3
 - getDispatchMethodS3, 4
 - getGenericS3, 5
 - getMethodS3, 5
 - setMethodS3, 7
- *Topic **package**
 - R.methodsS3-package, 2
- *Topic **programming**
 - findDispatchMethodsS3, 3
 - getDispatchMethodS3, 4
 - getGenericS3, 5
 - getMethodS3, 5
 - setMethodS3, 7

- character, 4
- class, 3, 4

- environment, 5
- Exception, 9

- findDispatchMethodsS3, 3, 4
- function, 4, 5, 8

- getDispatchMethodS3, 4, 4
- getGenericS3, 5, 6
- getMethodS3, 5, 5
- getS3method, 6

- isGenericS3, 5

- list, 4, 8

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

- setGenericS3, 5
- setMethodS3, 2, 6, 7

- throw, 9, 9

- TRUE, 4, 7, 8
- tryCatch, 9

- UseMethod, 8

- vector, 3, 4

- warning, 7