Package 'Xmisc'

February 19, 2015

Version 0.2.1

Date 2014-08-12 09:55:10 EDT

Title Xiaobei's miscellaneous classes and functions

Description This is Xiaobei's miscellaneous classes and functions useful when developing R packages, particularly for OOP using R Reference Class.

License GPL (>= 2)

LazyLoad TRUE

NeedsCompilation no

URL http://CRAN.R-project.org/package=Xmisc

Repository CRAN

Depends R (>= 3.1.0)

Imports methods

Suggests RUnit, datasets, data.table

Enhances

Collate 'Xmisc-package.R' 'imports.R' 'util.R' 'list.R' 'internal.R' 'generic.R' 'xrefclass.R' 'valueparser.R' 'argumentparser.R' 'logging.R' 'unittest.R'

Author Xiaobei Zhao [aut, cre, cph]

Maintainer Xiaobei Zhao <xiaobei@binf.ku.dk>

Date/Publication 2014-08-12 22:38:09

R topics documented:

`Xmisc-package` *Xiaobei's miscellaneous classes and functions*

Description

This is Xiaobei's miscellaneous classes and functions useful when developing R packages, particularly for OOP using R Reference Class.

Details

Listed are a few highlighted features that may be of common interests.

- `xRefClass-class`
- `ArgumentParser-class`
- `logme`
- `lprintf`

You may find the latest version of `Xmisc` at `http://cran.r-project.org/web/packages/Xmisc/index.html`.

Author(s)

Xiaobei Zhao

References

Xiaobei Zhao (2014)

Argument-class *Argument*

Description

Argument

Fields

`name` character
`value` ANY
`type` character
`default` ANY
`help` character

Author(s)

Xiaobei Zhao

ArgumentParser-class *Parser for command-line options and arguments*

Description

Parser for command-line options and arguments

Fields

`cmdargs` list
`exeargs` character
`args` list
`types` list
`defaults` list
`helps` list
`usage` character
`description` character

Methods

`add_argument(name, ..., type, default, required = FALSE, help = "", dest, action)` Add an argument.
`add_description(x)` Add a description.
`add_usage(x)` Add a usage.
`make_help()` Make and display 'usage'.

Author(s)

Xiaobei Zhao

Examples

```
## Test
require(Xmisc)
parser <- ArgumentParser$new()
parser$add_argument('--a_str',type='character')
parser$add_argument('--b_num',type='numeric',default='0')
a_str
## character(0)
b_num
## [1] 0
message(parser$get_help())
## Usage:
##     /bin/exec/R ...
## Description:
## Options:
##     a_str character
##     b_num numeric   [ 0 ]

## Not run:
## Test from a command line
R -q -e "
require(methods);require(Xmisc);
parser <- ArgumentParser$new();
parser$add_argument('--a_str',type='character');
parser$add_argument('--b_num',type='numeric',default='0');
printme(a_str);printme(b_num);parser$get_help();
" --args --a_str='Hello World!' --b_num=1
## Loading required package: Xmisc
## ## a_str ##
## [1] "Hello World!"
## ## b_num ##
## [1] 1
## ...

## End(Not run)
```

---

as.loglevel *Coerces an object to loglevel*

Description

Coerces an object to `loglevel`

Usage

```
as.loglevel(x, loglevels = get_loglevel())
```

Arguments

x	object
loglevels	the defined loglevels

Value

loglevel

Author(s)

Xiaobei Zhao

---

atos *Convert an R object to a string*

Description

Convert an R object to a string

Usage

```
atos(x, envir = sys.frame(sys.parent(0)))
```

Arguments

x	an R object.
envir	the environment to use.

Value

character

Author(s)

Xiaobei Zhao

cat0 *Cat without space but with a newline at the end*

Description

Cat without space but with a newline at the end by default

Usage

```
cat0(..., file = "", sep = "", fill = FALSE, labels = NULL,
  append = FALSE)
```

Arguments

...	see cat
file	see cat
sep	see cat
fill	see cat
labels	see cat
append	see cat

Value

see cat

Author(s)

Xiaobei Zhao

character_to_logical *Convert a character string to logical.*

Description

Convert a character string to logical.

Usage

```
character_to_logical(x, ignore.case = TRUE)
```

Arguments

x	character
ignore.case	logical, whether case is ignored

Value

logical. TRUE for "y","yes","t","true" and "1"; FALSE for "n","no","f","false" and "0".

Author(s)

Xiaobei Zhao

Examples

```
character_to_logical("yes")
try(character_to_logical("hi"))
```

`check.packages` *Check if a package can be loaded*

Description

Check if a package can be loaded. If TRUE, load it as long as it has not yet been loaded.

Usage

```
check.packages(x, envir = sys.frame(sys.parent(0)), character.only = FALSE)
```

Arguments

`x`	package, see `library` or `require`.
`envir`	the `environment` to use.
`character.only`	see `library` or `require`.

Value

logical, whether a package can be loaded.

Author(s)

Xiaobei Zhao

See Also

`is.package.loaded`

Examples

```
check.packages("Xmisc")
check.packages(Xmisc)
x <- "Xmisc"
check.packages(x,character.only=TRUE)
```

dfchunk *Chunk data.frame into parts*

Description

Chunk data.frame into parts

Usage

```
dfchunk(x, n, balance.size = TRUE, balance.order = FALSE)
```

Arguments

x	data.frame or matrix
n	numeric, the number of chunks
balance.size	logical, see vchunk
balance.order	logical, see vchunk

Value

a list of data.frame

Author(s)

Xiaobei Zhao

Examples

```
dfchunk(iris,n=5)
dfchunk(iris[1:20,],n=3)
dfchunk(iris[1:20,],n=3,balance.order=TRUE)
```

dfconcat *Concatenate data.frame into a string*

Description

Concatenate data.frame into a string

Usage

```
dfconcat(x, sep = " ", ...)
```

Arguments

x	data.frame or matrix
sep	character, a delimiter
...	further arguments passed to ‘format‘. See format

Value

data.frame

Author(s)

Xiaobei Zhao

---

dfsort *Sort data.frame given levels of one column*

Description

Sort data.frame given levels of one column

Usage

```
dfsort(x, which.col, levels)
```

Arguments

x	data.frame
which.col	column index or name
levels	character see base::factor

Value

data.frame

Author(s)

Xiaobei Zhao

Examples

```
data(CO2)
dfsort(CO2,"Treatment",c("nonchilled","chilled"))
dfsort(CO2,3,c("chilled","nonchilled"))
```

dfsplit *Split data.frame given one leveled column*

Description

Split data.frame given one leveled column

Usage

```
dfsplit(x, which.col, levels)
```

Arguments

x	data.frame
which.col	column index or name
levels	character see base::factor

Value

named list

Author(s)

Xiaobei Zhao

Examples

```
x <- read.table(textConnection("
chr1  0   100
chr2  100 200
chr10 200 300
"),col.names=c('chr','start','end'))

## compare the results by base::split and dfsplit
split(x,f=x[,'chr'])
## $chr1
##    chr start end
## 1 chr1     0 100

## $chr10
##     chr start end
## 3 chr10   200 300

## $chr2
##    chr start end
## 2 chr2   100 200

dfsplit(x,'chr',c('chr1','chr2','chr10'))
## $chr1
```

```
##    chr start end
## 1 chr1     0 100

## $chr2
##    chr start end
## 2 chr2   100 200

## $chr10
##     chr start end
## 3 chr10   200 300
```

`dir.exists` *Does the directory exist*

Description

Does the directory exist

Usage

```
dir.exists(x)
```

Arguments

`x`	character, a directory name.

Value

logical

Author(s)

Xiaobei Zhao

`endswith` *Determine if a character string "ends with" specified characters*

Description

Determine if a character string "ends with" specified characters

Usage

```
endswith(x, char, ignore.case = FALSE)
```

Arguments

x	character, a string
char	character to match
ignore.case	logical, whether case is ignored

Value

logical

Author(s)

Xiaobei Zhao

Examples

```
endswith('Hello World','world',ignore.case=TRUE)
```

---

func *Funciton with attributes*

Description

Funciton with attributes (name, package)

Usage

```
func(x, name, package)
```

Arguments

x	function
name	character, the function name
package	character, where the function is from

Value

named funciton

Author(s)

Xiaobei Zhao

Examples

```
## Not run:
func(lm,'lm','stats')

## End(Not run)
```

`getone,list-method` *Get an element by index or name from a list*

Description

Get an element by index or name from a list

Usage

```
## S4 method for signature 'list'
getone(obj, x, safe = TRUE, msg = TRUE)
```

Arguments

`obj`	list
`x`	index or name of an element
`safe`	whether allow to get an element by index, in case this element is after any named element
`msg`	whether print a message

Value

list

Author(s)

Xiaobei Zhao

Examples

```
ll <- list(11,22,33,a=44,b=55,66,77,c=88,99)
getone(ll,numeric())
getone(ll,"a")
getone(ll,"c")
getone(ll,1)
getone(ll,7)
getone(ll,7,safe=FALSE)
```

getone-methods *getone-methods*

Description

getone-methods

Usage

```
getone(obj, ...)
```

Arguments

obj	object
...	further arguments

get_executable *Get the executable file path of a package*

Description

Get the executable file path of a package

Usage

```
get_executable(pkg, name = tolower(pkg), dir = "bin", mustWork = TRUE)
```

Arguments

pkg	character, the name of a package
name	character, the name of the executable file
dir	character, the directory in the package hierarchy
mustWork	See system.file

Value

character, the executable file path

Author(s)

Xiaobei Zhao

Examples

```
## Not run:
try(get_executable('Xmisc','Xmisc-argumentparser.R'))

## End(Not run)
```

get_extdata *Get the extdata file path of a package*

Description

Get the extdata file path of a package

Usage

```
get_extdata(pkg, name, dir = "extdata", mustWork = TRUE)
```

Arguments

pkg	character, the name of a package
name	character, the name of the extdata file
dir	character, the directory in the package hierarchy
mustWork	See system.file

Value

character, the extdata file path

Author(s)

Xiaobei Zhao

Examples

```
## Not run:
try(get_extdata('datasets','morley.tab','data'))

## End(Not run)
```

get_loglevel *Define log levels*

Description

Define log levels

Usage

```
get_loglevel()
```

Value

the defined loglevels

Author(s)

Xiaobei Zhao

`is.activeBindingFunction` *General test of a class name being activeBindingFunction*

Description

General test of a class name being activeBindingFunction

Usage

```
is.activeBindingFunction(cls)
```

Arguments

`cls`	the name of a class

Value

logical

Author(s)

Xiaobei Zhao

`is.connection` *Is a connection*

Description

Is a connection

Usage

```
is.connection(x)
```

Arguments

`x`	R object

Value

logical

Author(s)

Xiaobei Zhao

Examples

```
is.connection(textConnection(LETTERS))
```

`is.dir` *Is it a directory*

Description

Is it a directory

Usage

```
is.dir(x)
```

Arguments

`x`	character, a directory name.

Value

logical

Author(s)

Xiaobei Zhao

is.file *Is it a file*

Description

Is it a file

Usage

```
is.file(x)
```

Arguments

x	character, a file name.

Value

logical

Author(s)

Xiaobei Zhao

is.linux *Is the OS Linux*

Description

Is the OS Linux

Usage

```
is.linux()
```

Value

logical

Author(s)

Xiaobei Zhao

is.loglevel *General test of an object being interpretable as loglevel*

Description

General test of an object being interpretable as `loglevel`

Usage

```
is.loglevel(x, loglevels = get_loglevel())
```

Arguments

`x`	object
`loglevels`	the defined loglevels

Value

logical

Author(s)

Xiaobei Zhao

is.package.loaded *Check if a package is loaded*

Description

Check if a package is loaded

Usage

```
is.package.loaded(x, envir = sys.frame(sys.parent(0)),
  character.only = FALSE)
```

Arguments

`x`	package, see `library` or `require`.
`envir`	the `environment` to use.
`character.only`	see `library` or `require`.

Value

logical

Author(s)

Xiaobei Zhao

See Also

check.packages

Examples

```
is.package.loaded(Xmisc)
is.package.loaded("Xmisc")
x <- "Xmisc"
is.package.loaded(x) #FALSE
is.package.loaded(x,character.only=TRUE) #TRUE
```

---

is.uninitializedField *General test of a class being uninitializedField*

Description

General test of a class being uninitializedField

Usage

```
is.uninitializedField(x)
```

Arguments

x	class

Value

logical

Author(s)

Xiaobei Zhao

See Also

methods::ReferenceClasses

is.windows *Is the OS Windows*

Description

Is the OS Windows

Usage

```
is.windows()
```

Value

logical

Author(s)

Xiaobei Zhao

List-class *A class inherited directly from envRefClass*

Description

A class inherited directly from envRefClass

Fields

`data` list, a base::list

Methods

`popmany(x)` Pop many by indexes.

`popone(x, warn = TRUE, error = TRUE)` Pop the one at the given index/position (or name) in the list, and return it. If no index is specified, obj$popone() removes and returns the last one in the list.

`removeone(x)` Remove the first matched element whose value is x. Display an error if it does not exist.

Author(s)

Xiaobei Zhao

logme *Log the name and the content of an R object*

Description

Log the name and the content of an R object given levels of logger

Usage

```
logme(x = NULL, prefix = NULL, logger = NULL,
  envir = sys.frame(sys.parent(0)))
```

Arguments

x	ANY, an R object.
prefix	the prefix to log.
logger	logging level, one of: NULL, 'INFO', 'DEBUG', 'WARNING', 'ERROR', 'CRITICAL'
envir	the environment to use.

Author(s)

Xiaobei Zhao

See Also

printme

Examples

```
## log an object
x1 <- 1:6
logme(x1)

## log according to logger levels
bar <- function(x,envir=sys.frame(sys.parent(0))){
  for (.logger in get_loglevel()) {
    if (is.null(.logger)) .prefix <- 'NULL' else .prefix <- .logger
    logme(x,prefix=.prefix,logger=.logger,envir=envir)
  }
}
options(logger='DEBUG')
bar(1:6) # print logs of level NULL, INFO and DEBUG
options(logger='ERROR')
bar(1:6) # print logs of level NULL, INFO, DEBUG, WARNING and ERROR
```

logsave *Log a 'save'*

Description

Log a 'save'

Usage

```
logsave(x, logger = NULL, envir = sys.frame(sys.parent(0)))
```

Arguments

x	ANY, an R object.
logger	see logme
envir	the environment to use.

Author(s)

Xiaobei Zhao

Examples

```
inFpath <- "mydir/mypath"
logsave(inFpath)
```

lprintf *String formatting given an environment*

Description

String formatting given an environment

Usage

```
lprintf(x, envir = sys.frame(sys.parent(1)))
```

Arguments

x	character, a string to format.
envir	the environment to use.

Value

character

Author(s)

Xiaobei Zhao

See Also

sprintf

Examples

```
a="fox";b="dog";
x <- 'The quick brown %(a)s jumps over the lazy %(b)s?
Or the quick brown %(b)s jumps over the lazy %(a)s?'

## format given the global environment
lprintf(x)
## [1] "The quick brown fox jumps over the lazy dog?
## Or the quick brown dog jumps over the lazy fox?"

## format given a local environment
myenv <- new.env()
local(
  {a="coyote";b="dog";},
  envir=myenv
)
lprintf(x,myenv)
## [1] "The quick brown coyote jumps over the lazy dog?
## Or the quick brown dog jumps over the lazy coyote?"
```

---

lstrip *Strip a string with given characters at the beginning (left end)*

Description

Strip a string with given characters at the beginning (left end)

Usage

```
lstrip(x, char = " ")
```

Arguments

x	character, a string.
char	character to trim.

Value

character

Author(s)

Xiaobei Zhao

`make.dir` *Make a directory recursively*

Description

Make a directory recursively

Usage

```
make.dir(x, mode)
```

Arguments

`x`	character, a directory name.
`mode`	the mode of the path, see `dir.create`

Author(s)

Xiaobei Zhao

Examples

```
## Not run:
if (character_to_logical(
  raw_input("Would you like to create a directory for testing
  at current working directory?",c('yes','no')))){
  ## make.dir('testdir','751') # uncomment it to let R create the directory
}

## End(Not run)
```

`popmany,list-method` *Remove and return many elements from a list*

Description

Remove and return many elements from a list

Usage

```
## S4 method for signature 'list'
popmany(obj, x)
```

Arguments

obj	list
x	indexes

Value

the elements and the R object is altered in place

Author(s)

Xiaobei Zhao

---

popmany-methods *popmany-methods*

Description

popmany-methods

Usage

```
popmany(obj, ..., envir)
```

Arguments

obj	object
...	further arguments
envir	the environment to use

---

popone,list-method *Remove and return an element from a list*

Description

Remove and return an element from an R object, given name or index. Default: the last element Note: the R object is altered in place, like Python List pop()

Usage

```
## S4 method for signature 'list'
popone(obj, x, warn = TRUE, error = TRUE)
```

Arguments

obj	an R object
x	name or index
warn	logical, whether warn at error
error	logical, whether stop at error

Value

the element and the R object is altered in place

Author(s)

Xiaobei Zhao

Examples

```
ll <- list(1,2,3,a=4,b=5,6,7,c=8,9)
popone(ll,"a")
popone(ll,1) ## remove the 1st in ll
popone(ll,1) ## remove the next (2nd of the origin)
try(popone(list(),"a"))
```

---

popone-methods *popone-methods*

Description

popone-methods

Usage

```
popone(obj, ..., envir)
```

Arguments

obj	object
...	further arguments
envir	the environment to use

printme *Print the name and the content of an R object*

Description

Print the name and the content of an R object

Usage

```
printme(x = NULL, prefix = NULL, envir = sys.frame(sys.parent(0)))
```

Arguments

x	ANY, an R object.
prefix	the prefix to print.
envir	the environment to use.

Author(s)

Xiaobei Zhao

See Also

logme

Examples

```
## print an object
x1 <- 1:6
printme(x1)

## print with a prefix
foo <- function(x,envir=sys.frame(sys.parent(0))){
  printme(x,match.call(),envir=envir)
  invisible()
}
foo(1:6)
```

R5.value.default *R5.value.default*

Description

R5.value.default

Usage

```
R5.value.default(type, default = "list")
```

Arguments

type	the type of an R object
default	the default value

Value

ANY

Author(s)

Xiaobei Zhao

Examples

```
require(Xmisc)
R5.value.default('character')
try(R5.value.default(NULL))
R5.value.default('environment')

R5.value.default('hclust')
R5.value.default('dendrogram')
R5.value.default('formula')
R5.value.default('lm')
```

R5.value.parse *R5.value.parse*

Description

R5.value.parse

Usage

```
R5.value.parse(value, type, default = "list")
```

Arguments

value	the value passed to the parameter
type	the type of an R object
default	the default value

Value

ANY

Author(s)

Xiaobei Zhao

Examples

```
R5.value.parse(NULL,'logical')
R5.value.parse(1,'logical')

R5.value.parse(NULL,'hclust')
R5.value.parse(NULL,'dendrogram')
R5.value.parse(NULL,'formula')
R5.value.parse(NULL,'lm')

R5.value.parse(NULL,'character')
R5.value.parse("",'character')
## [1] ""
```

---

raw_input *Input from the terminal (in interactive use)*

Description

Input from the terminal (in interactive use), confined by choice if provided.

Usage

```
raw_input(msg = "", choice, strip = TRUE)
```

Arguments

msg	character, a message to input
choice	character, choices to confine the input
strip	logical, whether to strip trailing spaces of the input

Value

character

Author(s)

Xiaobei Zhao

Examples

```
## Not run:
raw_input("Please enter user name: ")
raw_input("Please confirm",choice=c("yes","no"))

## End(Not run)
```

---

removeone,list-method *Remove an element from an R object*

Description

Remove an element from an R object Note: the R object is altered in place

Usage

```
## S4 method for signature 'list'
removeone(obj, x, warn = TRUE, error = TRUE)
```

Arguments

obj	an R object
x	an element
warn	logical, whether warn at error
error	logical, whether stop at error

Value

NULL and the R object is altered in place

Author(s)

Xiaobei Zhao

Examples

```
ll=list(1,2,3,a=4,b=5,6,7,c=8,9)
removeone(ll,3)
```

removeone-methods *removeone-methods*

Description

removeone-methods

Usage

```
removeone(obj, ..., envir = sys.frame(sys.parent(2)))
```

Arguments

obj	object
...	further arguments
envir	the environment to use

rstrip *Strip a string with given chars at the (right) end*

Description

Strip a string with given chars at the (right) end

Usage

```
rstrip(x, char = " ")
```

Arguments

x	character, a string.
char	character to trim.

Value

character

Author(s)

Xiaobei Zhao

schunk *Chunk a string into parts*

Description

Chunk a string into parts

Usage

```
schunk(x, size, brk = "-", indent.width1 = 0,
  indent.width = indent.width1, concat = TRUE)
```

Arguments

x	character, a string to chunk.
size	numeric, the size of a chunk.
brk	character to link broken words.
indent.width1	numeric, indent of the first line
indent.width	numeric, indent of the other lines
concat	logical, whether to concatenate by a ‘newline‘

Value

character

Author(s)

Xiaobei Zhao

Examples

```
x <- 'The quick brown fox jumps over the lazy dog.'
cat(schunk(x,15),'\n')
cat(schunk(x,15,indent.width1=4),'\n') # indent all lines
cat(schunk(x,15,indent.width=4),'\n')  # indent lines other than the first
x <- 'The word, honorificabilitudinita, occurs in Shakespeare\'s
play Love\'s Labour\'s Lost, and means "with honorablenesses".'
cat(schunk(x,30),'\n')
## The word, honorificabilitudini-
## ta, occurs in Shakespeare's
## play Love's Labour's Lost, and
##  means "with honorablenesses".
```

srep *Replicate and concatenate a string*

Description

Replicate and concatenate a string

Usage

```
srep(x, ...)
```

Arguments

x	See rep
...	See rep

Value

character

Author(s)

Xiaobei Zhao

Examples

```
srep("*",5)
```

stampme *Print a message with a time stamp*

Description

Print a message with a time stamp

Usage

```
stampme(x)
```

Arguments

x	ANY, an R object.

Author(s)

Xiaobei Zhao

Examples

```
stampme('Hello World!')
```

`stampmsg` *Generate a diagnostic message from its arguments, with timestamp*

Description

Generate a diagnostic message from its arguments, with timestamp

Usage

```
stampmsg(..., domain = NULL, appendLF = TRUE)
```

Arguments

`...`	see `message`
`domain`	see `message`
`appendLF`	see `message`

Author(s)

Xiaobei Zhao

Examples

```
stampmsg(LETTERS)
```

`startswith` *Determine if a character string "starts with" specified characters*

Description

Determine if a character string "starts with" specified characters. A modified version of gdata::startsWith.

Usage

```
startswith(x, char, ignore.case = FALSE)
```

Arguments

`x`	character, a string.
`char`	character to match.
`ignore.case`	logical, whether case is ignored

Value

logical

Author(s)

Xiaobei Zhao

Examples

```
startswith('Hello World','hello',ignore.case=TRUE)
```

---

strip *Strip a string with given chars at both ends*

Description

Strip a string with given chars at both ends

Usage

```
strip(x, char = " ")
```

Arguments

x	character, a string.
char	character to trim.

Value

character

Author(s)

Xiaobei Zhao

strsplit.first *Split a string at the first 'split'*

Description

Split a string at the first 'split'

Usage

```
strsplit.first(x, split, ...)
```

Arguments

x	character, a string to split.
split,	see strsplit
...,	see strsplit

Value

list

Author(s)

Xiaobei Zhao

Examples

```
strsplit.first('inFpath="a=1.b=2.c=TRUE"',split="=")
```

Sys.Epoch *Get system epoch*

Description

Get system epoch

Usage

```
Sys.Epoch()
```

Value

numeric

Author(s)

Xiaobei Zhao

UnitTest-class *Unit testing for developing R packages*

Description

Unit testing for developing R packages

Details

Unit testing for developing R packages

Fields

`pkg` character the name of the package

`testDpath` character the absolute directory names where to look for test files. Default: <pkg>/tests

`testFnameRegexp` character Regular expression for matching test file names. Default: *.R

`testFuncRegexp` character Regular expression for matching test functions. Default: test.*

Author(s)

Xiaobei Zhao

Examples

```
## Not run:
pkg <- 'Xmisc'
test.obj <- UnitTest$new(pkg=pkg)
test.obj$runme()

## End(Not run)
```

valid.arg.index *Check validity of an index*

Description

Check validity of an index of a `list` object.

Usage

```
valid.arg.index(obj, x, safe = TRUE)
```

Arguments

obj	list
x	index or name of an element
safe	whether safe if an index is higher than the one of any named element

Value

numeric, the index. Return numeric(0) if the name or the index does not exist or when the index is invalid. If safe is FALSE, any index is valid; if safe is TRUE, an index is invalid when the indexed element is positionally after another named element.

Author(s)

Xiaobei Zhao

Examples

```
ll <- list(11,12,13,a=14,b=15,16,17,c=18,19)
valid.arg.index(ll,-1) # non-existing index
valid.arg.index(ll,0)  # non-existing index
valid.arg.index(ll,1)  # valid index
valid.arg.index(ll,2)  # valid index
valid.arg.index(ll,5)  # invalid index
valid.arg.index(ll,10) # non-existing index
valid.arg.index(ll,"a")# valid name
valid.arg.index(ll,"e")# non-existing name
valid.arg.index(ll,5,safe=FALSE) # still return the index
```

---

valid.mode *Return a valid mode given digits*

Description

Return a valid mode given digits

Usage

```
valid.mode(mode, digits = 4)
```

Arguments

mode	character, the mode of the path, see dir.create.
digits	numeric, either 3 or 4.

Value

mode

Author(s)

Xiaobei Zhao

Examples

```
valid.mode("777",4)
valid.mode("0777",3)
```

ValueParser-class *Parser for values*

Description

Parser for values

Fields

`value` ANY

`type` character

Author(s)

Xiaobei Zhao

Examples

```
ValueParser$new(value="",type='character')$get_value()
ValueParser$new(type='character')$get_value()
```

---

vchunk *Chunk a vector into parts*

Description

Chunk a vector into parts given the number of chunks or the max size of a chunk

Usage

```
vchunk(x, n = NULL, max.size = NULL, balance.size = TRUE,
  balance.order = FALSE)
```

Arguments

x	vector to chunk
n	numeric, the number of chunks
max.size	numeric, the maximal size of a chunk
balance.size	logical, as equal as possible. Whether return balanced chunks.
balance.order	logical, whether to balance the elements. Force balance.size to be TRUE. given their original orders.

Value

list

Author(s)

Xiaobei Zhao

Examples

```
vchunk(1:7,7)
vchunk(1:19,n=3)
vchunk(1:19,max.size=9) # size-balanced
vchunk(1:19,max.size=9,balance.size=FALSE) # size/order-unbalanced
vchunk(1:19,max.size=9,balance.size=FALSE,balance.order=TRUE) # order-balanced
vchunk(1:19,max.size=9,balance.order=TRUE) # size/order-balanced
```

---

vconcat *Concatenate vector into a string*

Description

Concatenate vector into a string

Usage

```
vconcat(x, sep = ", ", capsule = FALSE, quote = FALSE)
```

Arguments

x	vector
sep	character, a delimiter
capsule	logical, weather to capsule with 'c()'
quote	logical, weather to surround elements by double quotes.

Value

vector

Author(s)

Xiaobei Zhao

Examples

```
cat(vconcat(head(letters),capsule=TRUE,quote=TRUE),'\n')
## c("a", "b", "c", "d", "e", "f")

cat(vconcat(head(letters),sep='-'),'\n')
## a-b-c-d-e-f
```

---

write.data.table *A wrapper of write.table*

Description

A wrapper of write.table with customized parameters and parsing

Usage

```
write.data.table(outFpath = "", x, append = FALSE, sep = "\t",
  quote = FALSE, row.names = FALSE, col.names = !append, logger = NULL,
  ...)
```

Arguments

outFpath	file, see write.table
x	see write.table
append	see write.table
sep	see write.table
quote	see write.table
row.names	see write.table
col.names	see write.table
logger	see logme
...	further arguments passed to 'write.table'. See write.table

Author(s)

Xiaobei Zhao

`xRefClass-class` *Extended Reference Class*

Description

The Extended Reference Class (xRefClass) inherits directly from `envRefClass`. Listed are some of its key features:

- Method `initialize` passes arguments by position or name.
- Method `copy2`, a modified version of `copy`, is tolerant to `activeBindingFunction` as fields.
- Method `update` updates a class instance's methods according to any update of the class.

Details

Extended Reference Class

Fields

`.index` named_numeric indexes of args

`.default` named_list default values of args

`.meta` named_list additional args (meta information)

`.envir` environment. Default: as.environment(.self)

`.tmp.list` list for temporary storage

`.out.list` list for outputting

Methods

`copy2(shallow = FALSE)` Modified version of 'copy' to allow 'activeBindingFunction' as fields.

`update(x)` Modify method definition without re-create the class instance. x: character, methods to be updated.

Author(s)

Xiaobei Zhao

See Also

`methods::ReferenceClasses`

Examples

```
## Not run:
MyClass <-
  setRefClass(
    "MyClass",
    list(
      x="numeric",
      y="numeric",
      z=function(){x+y}
      ),
    contains="xRefClass",
    methods=list(
      initialize=function(...){
        .idx <- c(x=1,y=2)
        callSuper(...,.index=.idx)
      },
      printme=function(){
        cat('Hello World!','\n')
      }
      )
    )

## Method initialize - pass by position
obj <- MyClass$new(1,2)
obj$x
obj$y

## Method initialize - pass by name
obj <- MyClass$new(y=2)
obj$x
obj$y

## Method copy
## obj <- MyClass$new(1,2)
## obk <- obj$copy()    # Fail!
## ## Error in (function ()  : unused argument (quote("myclass"))

## Method copy2
obj <- MyClass$new(1,2) # No such error!
obk <- obj$copy2()
obk$z

## Method update
obj <- MyClass$new()
obj$printme()
MyClass <- # To modify one of the original functions
  setRefClass(
    "MyClass",
    list(
      x="numeric",
      y="numeric",
      z=function(){x+y}
```

```
  ),
 contains="xRefClass",
 methods=list(
  initialize=function(...){
   .idx <- c(x=1,y=2)
   callSuper(...,.index=.idx)
  },
  printme=function(){ # This function is modified
   cat('Hello R!','\n')
  }
  )
 )
obj$printme() # The function is yet not modified
## Hello World!
obj$update("printme") # update the function
obj$printme() # The function is modified
## Hello R!

## End(Not run)
```

Index