Package 'ParamSetI'

January 4, 2019

Title Manage parameter tag/value pairs as an inheritable ReferenceClass object

Version 1.17.0

Date 2018-12-18

Author Charles Tilford [aut, cre]

Maintainer Charles Tilford <cran@agent.fastmail.com>

Description This package is designed to be inherited by other ReferenceClass objects. It manages key/value pairs that hold parameter information for the parent object.

URL https://github.com/maptracker/setfisher/tree/master/ParamSetI

BugReports https://github.com/maptracker/setfisher/issues

Depends R (>= 3.1)

Imports methods, EventLogger (>= 1.15.0), CatMisc (>= 1.0.0)

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests testthat

NeedsCompilation no

R topics documented:

allParams *All Parameters*

Description

ParamSetI object method to list all parameter keys

Arguments

help	Default FALSE. If TRUE, show this help and perform no other actions.

Details

```
## Method Usage:
myObject$allParams( help=TRUE )

myObject$allParams()
```

Lists all "known" parameter key names. This includes parameters that have been set, as well as those that are unset but have been described.

Value

A character vector of parameter key names

See Also

hasParam, showParameters, param

Examples

```
## This demo defines a toy object inheriting ParamSetI:
demo("exampleParamSetObject", package="ParamSetI", ask=FALSE)
pse <- ParamSetExample( params=list(inc=10L) )

pse$allParams()
pse$param("puppy", "fuzzy") # Add a new parameter
pse$allParams()
```

defineParameters *Define Parameters Set Param List*

Description

ParamSetI object method to bulk set definitions (descriptions) for parameters

Arguments

x	Required, the block of text to parse (character)
help	Default FALSE. If TRUE, show this help and perform no other actions.

Details

```
## Method Usage:
myObject$defineParameters( help=TRUE )

myObject$defineParameters( x )
```

A mechanism to bulk set parameters using a block of text. The text block will be split into lines and parsed with regular expressions to extract names, class restrictions and definitions. The method does not define values.

See Also

param, setParamList, showParameters

Examples

```
## This demo defines a toy object inheriting ParamSetI:
demo("exampleParamSetObject", package="ParamSetI", ask=FALSE)
pse <- ParamSetExample( params=list(inc=10L) )

# Define four new parameters, three of which have class
# restrictions, one does not:
pse$defineParameters("
   color [character] The color of the object, a name or hex value
  weight [numeric]   The mass of the object, in kilograms
 inStock [logical]   Flag to indicate if inventory is available
    misc Random information, see Jacob in logistics for more info
")

# Set a few of these
pse$param("weight", 74.3)
pse$param("instock", "yes") # Oops, violates class restriction
pse$param("misc", "[/][Stack on lower shelf/No hooks]")

# Show all the definitions
pse$showParameters( na.rm=FALSE )
```

---

hasParam *Has Parameter*

Description

ParamSetI object method to test if an object has a parameter or not

Arguments

key	Default NULL. One or more parameter names
help	Default FALSE. If TRUE, show this help and perform no other actions.

Details

```
## Method Usage:
myObject$hasParam( help=TRUE )

myObject$hasParam( key=NULL )
```

Checks if the provided key(s) are defined in the parameter set. Case/capitalization is ignored.

Value

A logical vector with TRUE indicating presence of the corresponding key in the parameter set

See Also

allParams

Examples

```
## This demo defines a toy object inheriting ParamSetI:
demo("exampleParamSetObject", package="ParamSetI", ask=FALSE)
pse <- ParamSetExample( params=list(inc=10L) )

pse$showParameters()
pse$hasParam("inc")
pse$hasParam("maximum operating depth")
```

`param` *Paramater*

Description

ParamSetI object method to get/set a parameter

Arguments

`key`	Default NA. The name of the parameter. If NA will simply return NA. Key names are case insensitive.
`val`	Default NA. Optional new value. If not NA, then the parameter value will be set, with behavior modified by some of the flags below
`append`	Default FALSE, which will cause the parameter to be set to `val`. If TRUE, `val` will be appended to the end of the current vector holding the value for `key`.
`default`	Default NA. Optional value to return if the value of `key` is 'not defined'; see is.def for values that are considered 'undefined'
`clobber`	Default TRUE, which allows an already-set value to be replaced with `val`. Using FALSE is primarily used for managing default settings.
`check.class`	Default NULL, which will check the parameter definitions and use any class found there. If the value is NA or '', then there will be no class check. Otherwise, is.class() will be tested with the provided class name against the provided `val` to see if it matches. If not, an error will be reported and `key` will not be set. The value 'percent' will be interpreted as 'numeric'.

`is.scalar`	Default NULL. If TRUE then only `val[1]` will be taken. If FALSE, then all elements of `val` to be assigned to `key`. If NULL, then if val is only a single element, but matches the pattern '[^]][.+]', then it will be split as a vector - see selfSplittingString
`coerce`	Default TRUE, which will attempt to coerce `val` to `check.class` in the event that check.class is not NA
`help`	Default FALSE. If TRUE, show this help and perform no other actions.

Details

```
## Method Usage:
myObject$param( help=TRUE )

myObject$param(key=NA, val=NA, append=FALSE, default=NA, clobber=TRUE,
               check.class=NULL, is.scalar=NULL, coerce=TRUE )
```

Gets or sets parameters on the object

Value

The value of the parameter, or NA if the key does not exist or is unset

See Also

showParameters, paramClass, is.def

Examples

```
## This demo defines a toy object inheriting ParamSetI:
demo("exampleParamSetObject", package="ParamSetI", ask=FALSE)
pse <- ParamSetExample( params=list(inc=10L) )

# Get a parameter
pse$param("inc")
# Set a parameter
pse$param("inc", 33L)

# Setting a parameter to an illegal class is prevented
pse$param("inc", "swamp monster")
# ... unless you take the safety off:
pse$param("inc", "swamp monster", check.class=FALSE)
# ... or specify an allowed class:
pse$param("inc", "swamp monster", check.class="character")

# coercable classes are allowed:
pse$param("inc", 42) # Numeric, not integer
# ... unless you don't allow it
pse$param("inc", 42, coerce=FALSE)

# Multiple values are normally ignored
pse$param("inc", 4:7)
# ... unless you indicate otherwise:
pse$param("inc", 4:7, is.scalar=FALSE)
# ... or you indicate that you want to append new values
pse$param("inc", 8:10, append=TRUE) # but only the first value!
```

```
# ... or you indicate *both*:
pse$param("inc", 11:13, is.scalar=FALSE, append=TRUE)
# ... or pass a string recognized by selfSplittingString
pse$param("inc", "[..][2..22..222]")

# clobber can be used to prevent overwriting a previously-set value
pse$param("inc", 1L, clobber=FALSE)

# You can specify a default value if none is already set
pse$param("color")
pse$param("color", default="purple")
```

`paramClass` *Parameter Class*

Description

ParamSetI object method to get/set class restrictions of a parameter

Arguments

`key`	Required, one or more parameter names
`val`	Default NULL. Optional character vector of new parameter class restrictions
`help`	Default FALSE. If TRUE, show this help and perform no other actions.

Details

```
## Method Usage:
myObject$paramClass( help=TRUE )

myObject$paramClass( key, val=NULL )
```

Parameters can have optional class restrictions (storage modes, eg 'integer' or 'logical') assigned to them. These serve as sanity checks when calling param; Unless `check.class` is set to FALSE, param will refuse to set a mismatched parameter value.

For setting large numbers of parameters, also consider defineParameters, which can parse a class restriction from a block of text, or setParamList, which can set parameters from a list and provide `check.class` as dots to param.

Value

A named character vector of class restrictions, with names corresponding to keys

See Also

defineParameters, showParameters

Examples

```
## This demo defines a toy object inheriting ParamSetI:
demo("exampleParamSetObject", package="ParamSetI", ask=FALSE)
pse <- ParamSetExample( params=list(inc=10L) )

# Get the class restriction for a parameter
pse$paramClass("inc")

# Set class restrictions for some new parameters
pse$paramClass(c("count", "isOnFire"),
               c("integer", "logical"))
# New class restrictions prevents illegal assignments
pse$param("isOnFire", "OMG YES")
pse$param("isOnFire", TRUE)
```

---

paramDefinition *Parameter Definition*

Description

ParamSetI object method to get/set the definition (description) of a parameter

Arguments

key	Required, one or more parameter names
val	Default NULL. Optional character vector of new parameter definitions
help	Default FALSE. If TRUE, show this help and perform no other actions.

Details

```
## Method Usage:
myObject$paramDefinition( help=TRUE )

myObject$paramDefinition( key, val=NULL )
```

Parameters can have optional descriptions to help inform users of their usage. This function returns these definitions, or allows them to be set. Deffinitions will be displayed in some output functions, such as showParameters.

For setting large numbers of parameters, also consider defineParameters, which can parse descriptions from a block of text.

Value

A named character vector of definitions, with names corresponding to keys

See Also

defineParameters, allParams, showParameters

Examples

```
## This demo defines a toy object inheriting ParamSetI:
demo("exampleParamSetObject", package="ParamSetI", ask=FALSE)
pse <- ParamSetExample( params=list(inc=10L) )

# Get the definition for a parameter
pse$paramDefinition("inc")
# Set the definitions for some new parameters
pse$paramDefinition(c("color", "velocity"),
                    c("The color of the object", "The speed of the object"))
# Show all descriptions
pse$paramDefinition( pse$allParams() )
```

`paramName` *Parameter Name*

Description

ParamSetI object method to get/set the pretty-print name of a parameter

Arguments

`key`	Required, one or more parameter names
`val`	Default NULL. Optional character vector of new parameter names. `val` can only differ from `key` in the capitalization, otherwise an error will be thrown.
`help`	Default FALSE. If TRUE, show this help and perform no other actions.

Details

```
## Method Usage:
myObject$paramName( help=TRUE )

myObject$paramName( key, val=NULL )
```

A mostly-aesthetic function to configure the case/capitalization of parameter names. Internally the package ignores case assigned to parameters. However, for display purposes the capitalization can be specifically set. This method allows a particular capitalization to be defined.

Value

A named character vector of class restrictions, with names corresponding to keys

See Also

defineParameters, showParameters

Examples

```
## This demo defines a toy object inheriting ParamSetI:
demo("exampleParamSetObject", package="ParamSetI", ask=FALSE)
pse <- ParamSetExample( params=list(inc=10L) )

# Get the pretty print name for a parameter
pse$paramClass("inc")

# That's boring. SHOUT IT
pse$paramClass("inc", "INC")

# Mostly handy for CamelCase. Note that key can be any case
pse$param("ADARKANDSTORMYNIGHT", "aDarkAndStormyNight")
```

ParamSetI-class *Paramter Set Interface*

Description

Inherited class holding parameter manipulation functions

Details

```
## Object creation
ParamSetI( help=TRUE ) # Show this help

## In general:
myPS <- ParamSetI( params=NA, paramDefinitions=NA, ...)

myPS$ANYMETHOD( help=TRUE ) # Each method has detailed help available
```

Fields

`paramSet` List holding key-value pairs used by an object

`paramDef` List holding parameter definitions and class restrictions

Methods

`allParams(help = FALSE)` Return all parameter keys, including those that are set or just described

`defineParameters(x, help = FALSE)` Bulk set parameter definitions with a block of text

`fieldDescriptions(help = FALSE)` A static list of brief descriptions for each field in this object

`hasParam(key = NULL, help = FALSE)` Test if one or more parameter keys exist in the object

`helpSections(help = FALSE)` Static list organizing object methods into conceptual sections

`initialize(verbose = TRUE, log = NULL, ...)` Create a new object using EventLogger()

`param(key = NA, val = NA, append = FALSE, default = NA, clobber = TRUE, check.cl` Get or set a parameter

`paramClass(key, val = NULL, help = FALSE)` Get or set class (storage mode) restrictions of parameters

`paramDefinition(key, val = NULL, help = FALSE)` Return the definitions, if any, for one or more parameter keys

`paramName(key, val = NULL, help = FALSE)` Set the capitalization to use for a parameter name

`setParamList(params = NULL, help = FALSE, ...)` Set multiple parameters using a list

`show(...)` A wrapper for showLog

`showParameters(na.rm = TRUE, help = FALSE)` Pretty-print parameter names, values and definitions

Examples

```
## Demo with inheritance of the class:
demo("objectInheritance", package="ParamSetI", ask=FALSE)
```

`selfSplittingString` *Self-Splitting String*

Description

Parses a string that defines a split token to turn it into a vector

Usage

```
selfSplittingString(x)
```

Arguments

`x`	Required, the string vector to proces

Details

Helper function for ParamSetI, takes a character vector as input. If the input is length 1 and is of format:

[<TOKEN>][<TEXT>]

... then <TEXT> will be split into a vector by <TOKEN>. For example:

[/][x/ y /z]

... becomes c("x", " y ", "z").

In all other cases, only a single value will be returned, corresponding to the first non-NA value in x.

If the parsed text includes " ## ", then it and the following text will be removed, and the following text will be attached as a 'comment' attribute

Value

NULL if passed NULL, otherwise a character vector

Examples

```
selfSplittingString(" [,][a,b, c ]  ")
# "a"   "b"   " c "

ltb <- selfSplittingString("[ and ][lions and tigers and bears] ## oh my")
str(ltb, "comment")
# "lions" "tigers" "bears"
# 'comment' attribute "oh my"

# A single return value (vector length 1) is enforced:
selfSplittingString(c("a","b","c"))
# "a"
```

`setParamList` *Set Param List*

Description

ParamSetI object method to set parameters in bulk using a list object

Arguments

`params`	Default NULL, which will be a no-op and return NA. Otherwise, a list holding the parameters, with the names representing key names and the contents the values to assign to each key
`...`	Any other parameters will be passed to param() for each key/value pair
`help`	Default FALSE. If TRUE, show this help and perform no other actions.

Details

```
## Method Usage:
myObject$setParamList( help=TRUE )

myObject$setParamList( params=NULL, ... )
```

A mechanism to bulk set parameters using a list. The list names will be taken as keys, and their contents as values.

See Also

defineParameters, param, showParameters

Examples

```
## This demo defines a toy object inheriting ParamSetI:
demo("exampleParamSetObject", package="ParamSetI", ask=FALSE)
pse <- ParamSetExample( params=list(inc=10L) )

# What is already set:
pse$showParameters()

# Reset some existing parameters:
pse$setParamList( list( inc=7L, dec=2L ) )
# Set some new parameters:
pse$setParamList( list(color="green", speed="88mph") )
# Check current values
pse$showParameters()

# Can unset clobber to set defaults without overwriting current values
pse$setParamList( list( inc=999L, dec=888L ), clobber=FALSE )
pse$showParameters()
```

`showParameters` *Show Parameters*

Description

ParamSetI object method to pretty-print parameters and their values

Arguments

`na.rm`	Default TRUE, which will exclude parameters that are not "defined" (see is.def for values that are considered 'undefined'). Setting to FALSE shows all parameters available from allParams.
`help`	Default FALSE. If TRUE, show this help and perform no other actions.

Details

```
## Method Usage:
myObject$showParameters( help=TRUE )

myObject$showParameters( na.rm=TRUE )
```

Show parameters in a format designed for human consumption. Parameters will be shown wrapped in a `$param()` call to illustrate how to access and change the parameter. If a definition has been set, it will be shown as a comment line under the parameter value.

See Also

param

Examples

```
## This demo defines a toy object inheriting ParamSetI:
demo("exampleParamSetObject", package="ParamSetI", ask=FALSE)
pse <- ParamSetExample( params=list(inc=10L) )

# Show all parameters with defined values
pse$showParameters(  )

# Show all parameters, including one that has an undefined value
# but an assigned deffinition (description)
pse$showParameters( na.rm=FALSE )
```

Index