Package 'often'

January 17, 2021

Title Schedules functions to run at periodic intervals

Version 0.1

Description Schedules functions to run at periodic intervals.

Depends R (>= 3.1.0), later, magrittr, methods

License GPL-3

LazyData true

ByteCompile true

RoxygenNote 7.1.1

R topics documented:

`addAction` *Switchboard objects*

Description

switchboards are chains of if-then statements, which can be very useful when long chains of such statements are needed, such as when programming bots. Using switchboards allows users to divide parts of their chain into different files or modules.

Usage

```
addAction(.self, condition, action, name = NULL, endChain = F)

delAction(.self, idx)

evaluateAll(.self)
```

Arguments

condition	A logical statement, to be evaluated when the entire switchboard is evaluated. If true, the associated action is executed. The content of this argument remains unevaluated and is converted to a language object, to be evaluated when the evaluate method is called. However, content that's placed between .() is evaluated when running this function.
action	A piece of code to be run when the associated if statement is evaluated and found to be true. Content placed within .() is evaluated when running this function, the rest remains unevaluated until the evaluateAll method is run.
name	A name for this particular if-then statement. Can be left empty.
endChain	Logical. Should this statement, if true, end the evaluation of any further statements?
idx	Character or numeric. The to-be-deleted if-then statement. You can provide either its name or its number.

Functions

- addAction: add an action to a switchboard.
- delAction: Delete an action from a switchboard.
- evaluateAll: Evaluate all actions in a switchboard. Evaluate all if-then statements in a switchboard.

Fields

actions A list of if-then statements.

---

makePeriodicValue *Make a periodically re-evaluated value-returning function.*

Description

Make a periodically re-evaluated value-returning function.

Usage

```
makePeriodicValue(fun, period, ...)
```

Arguments

fun	Function that generates a value.
period	Period, inseconds, after which to refresh the value.
...	Other arguments to be passed to the value-generating function.

Value

A function that returns a value, provided by the value-generating function, and refreshes it after the provided refresh period.

tableEnv *Schedule tasks*

Description

The following functions allow users to schedule functions in a virtual agenda, and initiate a loop that periodically checks whether any particular function is due.

Usage

```
tableEnv()

addJob(code, runtime, loopid = "main")

delJob(jobid, jobnum, status, delete = TRUE)

jobList()

setLoop(
  loopid = "main",
  rate = 1,
  tolerance = 60,
  on.error = "continue",
  on.miss = "continue",
  on.complete = "terminate"
)

startLoop(loopid = "main")

stopLoop(loopid = "main")
```

Arguments

code	Code of any length. It will remain unevaluated until the job is run - save for the parts of the code wrapped in `.()` brackets.
runtime	a future timestamp (use in conjunction with `now()` for convenience)
loopid	A handle to an event loop.
jobid	one or multiple jobids, which serve as identifiers for the scheduled jobs, to be deleted
jobnum	Number of job(s) to be deleted
status	status of job(s) to be deleted
delete	Logical. If true, deletes jobs from schedule. If false, sets their status to 'disabled' instead. Default is TRUE.
rate	numeric, in seconds. How often should the loop check for a new task to be run?
tolerance	numeric, in seconds; jobs that were scheduled this many seconds in the past will still be run. Jobs beyond this point will be considered missed.
on.error	character; what is to be done when a job results in an error? Defaults to "continue".

on.miss	character; what is to be done when a job is missed? Defaults to "continue".
on.complete	character; what is to be done to the loop when there are no more jobs to be run? Defaults to "terminate".

Details

A job can be any line(s) of code, and it will be run at the designated timstamp if the associated loop is running.

startLoop() starts a loop.

stopLoop() stops a loop.

Value

delJob() returns TRUE if successful, FALSE if failed.

Functions

- tableEnv: Returns the environment containing the looptable and the jobtable, allowing for direct modification rather than through helper functions.
- addJob: Add a job to the schedule.
- delJob: Delete a job. Use in conjunction with jobList(). Deletes job(s) based on their number in the job queue/schedule, their IDs, or their status.
- jobList: Returns a data.frame containing all jobs, finished and unfinished.
- setLoop: Configure an existing loop or create a new loop with custom settings. When started using startLoop(), this loop will periodically check for jobs to execute. When an unknown loopid is given, a new loop is created.
- startLoop: Start a loop with given loopid.
- stopLoop: Stop a running loop.

Examples

```
addJob(code=print("Hello World"),runtime=now()+5)
delme<-addJob(code=print("Goodbye World"),runtime=now()+6)
delJob(jobid=delme)
addJob(code=stopLoop(),runtime=now()+7)
setLoop(rate=0.5)
print(jobList())
startLoop()
```

Utilities *Utilities Useful tools for programming.*

Description

Utilities Useful tools for programming.

Usage

```
now()

bquote.arg(x)

check.types(...)

quantize(x, init, step, bias = c("round", "floor", "ceiling"))
```

Arguments

x	a vector of values
...	named arguments, where the name of the argument is a variable class, and the argument itself is the value to be checked for its type. function cannot be provided as argument name, so the shorthand fun must be used.
init	an initial value
step	steps over which to be quantized
bias	when quantizing, should the value be rounded up or down?

Details

now() returns the current numeric Linux timestamp.

bquote.arg() is to be used inside functions. It returns the content of the argument as a call, but evaluates parts of the argument that were wrapped inside .(), akin to bquote().

check.types() checks the class of its arguments, and errors if an argument does not match the class that was provided as argument name.

quantize() quantizes a vector.

Value

quantize() returns a quantized numeric vector.

Examples

```
now()
testfunction<-function(x){ bquote.arg(x) }
testfunction(a + b)
# a + b
testfunction(a + b + .(5+5))
# a + b + 10
check.types(numeric=3,character="a",fun=read.csv)
# TRUE
## Not run:
check.types(numeric="3")

## End(Not run)
# Error in check.types(numeric = "3") :
# Argument 1 with value 3 should be of type numeric, but is of type character
quantize(1:20,0.5,2,"ceiling")
# 2.5  2.5  4.5  4.5  6.5  6.5  8.5  8.5 10.5 10.5 12.5 12.5 14.5 14.5 16.5 16.5 18.5 18.5 20.5 20.5
```

Index