Package 'log4r'

September 29, 2014

Encoding UTF-8

Type Package

Title A simple logging system for R, based on log4j.

Version 0.2

Date 2014-09-29

Description logr4 provides an object-oriented logging system that uses an API roughly equivalent to log4j and its related variants.

License Artistic-2.0

LazyLoad yes

Suggests testthat

URL https://github.com/johnmyleswhite/log4r

BugReports https://github.com/johnmyleswhite/log4r/issues

R topics documented:

log4r-package *A simple logging system for R, based on log4j.*

Description

logr4 provides an object-oriented logging system that uses an API roughly equivalent to log4j and its related variants.

Details

Package:	log4r
Type:	Package
Version:	0.2
Date:	2014-09-29
License:	Artistic-2.0
LazyLoad:	yes

Maintainer: Kirill Müller <krlmlr+r@mailbox.org>

URL: https://github.com/johnmyleswhite/log4r

Issue tracker: https://github.com/johnmyleswhite/log4r/issues

References

See the log4j documentation or the documentation for its many derivatives to understand the origins of this logging system.

Examples

```
# Import the log4r package.
library('log4r')

# Create a new logger object with create.logger().
logger <- create.logger()

# Set the logger's file output.
logfile(logger) <- 'base.log'

# Set the current level of the logger.
level(logger) <- 'INFO'

# Try logging messages with different priorities.
# At priority level INFO, a call to debug() won't print anything.
debug(logger, 'A Debugging Message')
info(logger, 'An Info Message')
warn(logger, 'A Warning Message')
error(logger, 'An Error Message')
```

```
fatal(logger, 'A Fatal Error Message')
```

`create.logger` *Creates a logger object.*

Description

Creates a logger object.

Usage

```
create.logger(logfile = "logfile.log", level = "FATAL", logformat = NULL)
```

Arguments

`logfile`	The full pathname of the file you want log messages to be written to.
`level`	The level at which the logger is initialized. Will be coerced using `as.loglevel`.
`logformat`	The format string used when writing messages to the log file.

See Also

`loglevel`, `level.logger`

Examples

```
library('log4r')

logger <- create.logger(logfile = 'debugging.log', level = "DEBUG")
```

`level` *Set or get the priority level for a logger object.*

Description

The priority level can be an integer from the set 1..5 (otherwise it will be modified sensibly to fit in that range), or a named logging level (one of "DEBUG", "INFO", "WARN", "ERROR", "FATAL"). An object of class loglevel is also accepted; other input will be coerced using `as.loglevel`.

Usage

```
level(x)

level(x) <- value

## S3 method for class 'logger'
level(x)

## S3 replacement method for class 'logger'
level(x) <- value
```

Arguments

x	An object of class logger.
value	A loglevel.

See Also

loglevel

Examples

```
library('log4r')

logger <- create.logger(logfile = 'debugging.log', level = 1)
level(logger)
level(logger) <- "FATAL"
```

---

levellog *Write messages to logs at a given priority level.*

Description

Write messages to logs at a given priority level.

Usage

```
levellog(logger, level, message)

debug(logger, message)

info(logger, message)

warn(logger, message)

error(logger, message)

fatal(logger, message)
```

Arguments

logger	An object of class 'logger'.
level	The desired priority level: a number, a character, or an object of class 'loglevel'. Will be coerced using as.loglevel.
message	A string to be printed to the log with the corresponding priority level.

See Also

loglevel

Examples

```
library('log4r')

logger <- create.logger(logfile = 'debugging.log', level = "WARN")

levellog(logger, 'WARN', 'First warning from our code')
debug(logger, 'Debugging our code')
info(logger, 'Information about our code')
warn(logger, 'Another warning from our code')
error(logger, 'An error from our code')
fatal(logger, 'I\'m outta here')
```

---

logfile *Get or set the logfile for a logger object.*

Description

Get or set the logfile for a logger object.

Usage

```
logfile(x)

logfile(x) <- value

## S3 method for class 'logger'
logfile(x)

## S3 replacement method for class 'logger'
logfile(x) <- value
```

Arguments

x	An object of class logger.
value	The path name of a file to be used for logging. Must be a valid path in an already existing directory

Examples

```
library('log4r')

logger <- create.logger()
print(logfile(logger))
logfile(logger) <- 'debug.log'
debug(logger, 'A Debugging Message')
```

`logformat` *Get or set the format string for a logger object.*

Description

Get or set the format string for a logger object.

Usage

```
logformat(x)

logformat(x) <- value

## S3 method for class 'logger'
logformat(x)

## S3 replacement method for class 'logger'
logformat(x) <- value
```

Arguments

`x`	An object of class logger.
`value`	A string containing a proper format string.

Examples

```
library('log4r')

logger <- create.logger(logfile = 'debugging.log', level = 'DEBUG')
print(logformat(logger))
logformat(logger) <- 'FORMAT STRING'
```

`loglevel` *Logging levels*

Description

Functions for handling logging levels. With each log entry, a logging level is associated that indicate its severity – debugging output, informational output, warning message, error message or fatal error. Each logger only prints log entries where the log level is equal or above its threshold.

Usage

```
loglevel(i)

is.loglevel(x, ...)

as.loglevel(i)

## S3 method for class 'loglevel'
print(x, ...)

## S3 method for class 'loglevel'
as.numeric(x, ...)

## S3 method for class 'loglevel'
as.character(x, ...)

available.loglevels()

verbosity(v)
```

Arguments

i	An integer from the set 1..5. Otherwise it will be modified sensibly to fit in that range. Alternatively, a named logging level (one of "DEBUG", "INFO", "WARN", "ERROR", "FATAL").
x	An object of class "loglevel"
...	Unused
v	A verbosity level from the set 5..1. For historical reasons, they do not match the log levels; a verbosity level of 1 corresponds to a logging level of 5, 2 corresponds to 4, etc.

Details

To specify a logging level, use a character value, e.g. `"WARN"`, or an integer between 1 and 5. The function `available.levels` lists all possible logging levels.

Value

An object of class "loglevel"

Examples

```
loglevel(2) == loglevel("INFO")
loglevel("WARN") < loglevel("ERROR")
loglevel(-1)
try(loglevel("UNDEFINED"))
is.loglevel("DEBUG")
is.loglevel(loglevel("DEBUG"))
```

```
as.numeric(loglevel("FATAL"))
available.loglevels()

## Not run:
library(optparse)
library(log4r)

optlist <- list(make_option(c('-v', '--verbosity-level'),
  type = "integer",
  dest = "verbosity",
  default = 1,
  help = "Verbosity threshold (5=DEBUG, 4=INFO 3=WARN, 2=ERROR, 1=FATAL)"))

optparser <- OptionParser(option_list=optlist)
opt <- parse_args(optparser)

my.logger <- create.logger(logfile = "", level = verbosity(opt$verbosity))

fatal(my.logger, "Fatal message")
error(my.logger, "Error message")
warn(my.logger, "Warning message")
info(my.logger, "Informational message")
debug(my.logger, "Debugging message")

## End(Not run)
```

`write.message` *A hidden function for handling the writing of logging messages.*

Description

A hidden function for handling the writing of logging messages.

Usage

```
write.message(logger, message)
```

Arguments

`logger`	An object of class logger.
`message`	A message to be written to the log file. The current timestamp will be appended to this message.

Index