Package 'reportr'

October 6, 2016

Version 1.2.2

Date 2016-10-06

Title A General Message and Error Reporting System

Author Jon Clayden

Maintainer Jon Clayden <code@clayden.org>

Imports ore

Suggests testthat (>= 0.11.0)

Description Provides a system for reporting messages, which provides certain useful features over the standard R system, such as the incorporation of output consolidation, message filtering, expression substitution, automatic generation of stack traces for debugging, and conditional reporting based on the current ``output level".

License GPL-2

URL https://github.com/jonclayden/reportr

BugReports https://github.com/jonclayden/reportr/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2016-10-06 19:04:49

R topics documented:

reportr *The reportr message reporting system*

Description

Functions for reporting informative messages, warnings and errors. These are provided as alternatives to the `message`, `warning` and `stop` functions in base R.

Usage

```
setOutputLevel(level)

getOutputLevel()

withReportrHandlers(expr)

ask(..., default = NULL, prefixFormat = NULL)

report(level, ..., prefixFormat = NULL)

flag(level, ...)

reportFlags()

clearFlags()
```

Arguments

`level`	The level of output message to produce, or for `setOutputLevel`, the minimum level to display. See Details.
`expr`	An expression to be evaluated.
`default`	A default return value, to be used when the message is filtered out or the output level is above `Question`.
`prefixFormat`	The format of the string prepended to the message. See Details.
`...`	Objects which can be coerced to mode `character`. These will be passed through function `es` (from the `ore` package) for expression substitution, and then printed with no space between them. Options to `es`, such as `round`, may also be given.

Details

The `reportr` system for reporting messages provides certain useful features over the standard R system, such as the incorporation of output consolidation, message filtering, expression substitution, automatic generation of stack traces for debugging, and conditional reporting based on the current "output level". Messages of level at least equal to the value of option `reportrStderrLevel` are written to standard error (`stderr`); others are written to standard output (`stdout`).

The output level is set by the `setOutputLevel` function, and governs whether a particular call to `report` will actually report anything. Output levels are described by the `OL` object, a list with components `Debug`, `Verbose`, `Info`, `Warning`, `Question`, `Error` and `Fatal`. Any call to `report` using a level lower than the current output level will produce no output. If `report` is called before `setOutputLevel`, the output level will default to `Info` (with a message).

The `flag` function is called like `report`, but it stores messages for later reporting, like `warning`, rather than reporting them immediately. Stored messages are reported when `report` is next called, at which point multiple instances of the same message are consolidated where possible. The user may also manually force stored messages to be reported by calling `reportFlags`, or remove them with `clearFlags`. Note that the output level at the time that `reportFlags` is called (implicitly or explicitly) will determine whether the flags are printed.

The `ask` function requests input from the user, using `readline`, at output level `Question`. The text argument then forms the text of the question, and `ask` returns the text entered by the user.

The call `report(Error,...)` is largely similar to `stop(...)` in most cases, except that a stack trace will be printed if the current output level is `Debug`. The "abort" restart is invoked in any case. No other standard conditions are signalled by `report`. Stack traces can be generated at lower output levels, if desired, by setting the `reportrStackTraceLevel` option.

The `withReportrHandlers` function evaluates `expr` in a context in which R errors, warnings and messages will be handled by reportr, rather than by the standard R functions.

The `prefixFormat` argument to `report` and `ask` controls how the output message is formatted. It takes the form of a `sprintf`-style format string, but with different expansions for percent-escapes. Specifically, `"%d"` expands to a series of stars indicating the current stack depth; `"%f"` gives the name of the function calling `report` or `ask`; `"%l"` and `"%L"` give lower and upper case versions of the level of the message, respectively; and `"%p"` expands to the ID of the current R process (see `Sys.getpid`). The default is `"%d%L: "`, giving a prefix such as `"* * INFO: "`, but this default can be overridden by setting the `reportrPrefixFormat` option.

A number of other options influence the output produced by reportr. `getOutputLevel` and `setOutputLevel` get and set the `reportrOutputLevel` option, which can be set directly if preferred. The options `reportrMessageFilterIn` and `reportrMessageFilterOut` can contain a single character string representing a Perl regular expression, in which case only messages which match (`reportrMessageFilterIn`) or do not match (`reportrMessageFilterOut`) the regular expression will be retained. Likewise, the `reportrStackFilterIn` and `reportrStackFilterOut` options filter the call stack.

Value

These functions are mainly called for their side effects, but `getOutputLevel` returns the current output level, `withReportrHandlers` returns the value of the evaluated expression, and `ask` returns a character vector of length one giving the user's response.

Author(s)

Jon Clayden

See Also

`es` (in package `ore`) for expression substitution (which is performed on messages). `message`, `warning`, `stop` and `condition` for the normal R message and condition signalling framework.

Examples

```
setOutputLevel(OL$Warning)
report(Info, "Test message")  # no output
setOutputLevel(OL$Info)
report(Info, "Test message")  # prints the message

flag(Warning, "Test warning")  # no output
flag(Warning, "Test warning")  # repeated warning
reportFlags()  # consolidates the warnings and prints the message

## Not run: name <- ask("What is your name?")
report(OL$Info, "Hello, #{name}")
## End(Not run)
```

Index