Package 'reportr'

January 27, 2015

Version 1.1.2

Date 2014-03-10

Title A general message and error reporting system

Author Jon Clayden

Maintainer Jon Clayden <code@clayden.org>

Depends R (>= 2.5.0)

Description The reportr package 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''.

LazyLoad yes

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2014-03-12 16:30:54

R topics documented:

`infix` *Test a character vector for a match against a regular expression*

Description

Test whether the elements of a character vector match, or don't match, a Perl regular expression.

Usage

```
X %~% Y
X %!~% Y
```

Arguments

X	A character vector to test for matches.
Y	A character vector of length one representing a Perl regular expression to match against.

Details

%~% is infix shorthand for `regexpr(Y,X,perl=TRUE) != -1`.

Value

For %~%, TRUE for each element of X which matches the regular expression described by Y, and FALSE for every other element. For %!~%, the opposite.

Author(s)

Jon Clayden

See Also

regexpr

Examples

```
c("foo", "foo bar") %~% "\\w+\\s+(\\w+)?"  # c(FALSE, TRUE)
c("foo", "foo bar") %!~% "\\w+\\s+(\\w+)?"  # c(TRUE, FALSE)
```

---

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

```
getOutputLevel()
setOutputLevel(level)

report(level, ..., prefixFormat = NULL)
flag(level, ...)
ask(..., default = NULL, prefixFormat = NULL)

reportFlags()
clearFlags()

withReportrHandlers(expr)
```

Arguments

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

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 Warning level and above 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 and Error; and 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 OL$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(OL$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

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

flag(OL$Warning, "Test warning")  # no output
flag(OL$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)
```

s *Simple expression substitution*

Description

Evaluate R expressions and substitute their values into a character vector.

Usage

```
s(strings, round = NULL, signif = NULL, envir = parent.frame())
```

Arguments

strings	A character vector containing the strings to substitute into, each containing any number of "#{}" blocks.
round	NULL or a single integer, giving the number of decimal digits for rounding numeric expressions. This argument takes priority over signif.
signif	NULL or a single integer, giving the number of significant decimal digits to use for numeric expressions. The round argument takes priority over this one and will be used if not NULL.
envir	The environment to evaluate expressions in.

Details

Each part of each string surrounded by "#{}" is extracted, evaluated as R code in the specified environment, and then its value is substituted back into the string. The literal string "#{}" can be obtained by escaping the hash character, viz. "\#{}". The block may contain multiple R expressions, separated by semicolons, but may not contain additional braces.

If the reportrNumericRepresentation option is defined and is a list, its round and signif elements provide defaults for the arguments of the same name.

Value

A character vector containing the final strings, with expressions substituted into them.

Author(s)

Jon Clayden

See Also

report

Examples

```
s("pi is #{pi}")
s("pi is \#{pi}")
s("The square-root of pi is approximately #{sqrt(pi)}", signif=4)
s("1/(1+x) for x=3 is #{x <- 3; 1/(1+x)}")
```

Index