Package 'evaluate'

July 2, 2014

Type Package

Title Parsing and evaluation tools that provide more details than the default.

Version 0.5.5

Date 2014-04-30

Description Parsing and evaluation tools that make it easy to recreate the command line behaviour of R.

License GPL

Depends R (>= 2.14.0)

Imports stringr (>= 0.6.2)

Suggests testthat, lattice, ggplot2

Author Hadley Wickham [aut],Yihui Xie [cre, ctb],Barret Schloerke [ctb]

Maintainer Yihui Xie <xie@yihui.name>

NeedsCompilation no

Repository CRAN

Date/Publication 2014-04-29 22:08:31

R topics documented:

`evaluate` *Evaluate input and return all details of evaluation.*

Description

Compare to `eval`, `evaluate` captures all of the information necessary to recreate the output as if you had copied and pasted the code into a R terminal. It captures messages, warnings, errors and output, all correctly interleaved in the order in which they occured. It stores the final result, whether or not it should be visible, and the contents of the current graphics device.

Usage

```
evaluate(input, envir = parent.frame(), enclos = NULL, debug = FALSE,
   stop_on_error = 0L, keep_warning = TRUE, keep_message = TRUE, new_device = TRUE,
    output_handler = default_output_handler)
```

Arguments

`input`	input object to be parsed an evaluated. Maybe a string, file connection or function.
`envir`	environment in which to evaluate expressions
`enclos`	when `envir` is a list or data frame, this is treated as the parent environment to `envir`.
`debug`	if `TRUE`, displays information useful for debugging, including all output that evaluate captures
`stop_on_error`	if `2`, evaluation will stop on first error and you will get no results back. If `1`, evaluation will stop on first error, but you will get back all results up to that point. If `0` will continue running all code, just as if you'd pasted the code into the command line.
`keep_warning,keep_message`	whether to record warnings and messages
`new_device`	if `TRUE`, will open a new graphics device and automatically close it after completion. This prevents evaluation from interfering with your existing graphics environment.
`output_handler`	an instance of `output_handler` that processes the output from the evaluation. The default simply prints the visible return values.

getSrcRegion *Extract a rectangular region of a srcfile*

Description

Extract a rectangular region of a srcfile

Usage

```
getSrcRegion(srcfile, x1, x2, y1, y2)
```

Arguments

srcfile	string
x1	start line
x2	end line
y1	start col
y2	end col

Value

a string

line_prompt *Line prompt.*

Description

Format a single expression as if it had been entered at the command prompt.

Usage

```
line_prompt(x, prompt = getOption("prompt"), continue = getOption("continue"))
```

Arguments

x	string representing a single expression
prompt	prompt for first line
continue	prompt for subsequent lines

Value

a string

new_output_handler *Custom output handlers.*

Description

An output_handler handles the results of evaluate, including the values, graphics, conditions. Each type of output is handled by a particular function in the handler object.

Usage

```
new_output_handler(source = identity, text = identity, graphics = identity,
    message = identity, warning = identity, error = identity, value = render)
```

Arguments

source	Function to handle the echoed source code under evaluation.
text	Function to handle any textual console output.
graphics	Function to handle graphics, as returned by recordPlot.
message	Function to handle message output.
warning	Function to handle warning output.
error	Function to handle stop output.
value	Function to handle the values returned from evaluation. If it only has one argument, only visible values are handled; if it has more arguments, the second argument indicates whether the value is visible.

Details

The handler functions should accept an output object as their first argument. The return value of the handlers is ignored, except in the case of the value handler, where a visible return value is saved in the output list.

Calling the constructor with no arguments results in the default handler, which mimics the behavior of the console by printing visible values.

Note that recursion is common: for example, if value does any printing, then the text or graphics handlers may be called.

Value

A new output_handler object

parse_all *Parse, retaining comments.*

Description

Works very similarly to parse, but also keeps original formatting and comments.

Usage

```
parse_all(x)
```

Arguments

x	object to parse. Can be a string, a file connection, or a function

Value

a data.frame with columns `src`, the source code, and `eval`

replay *Replay a list of evaluated results.*

Description

Replay a list of evaluated results, as if you'd run them in an R terminal.

Usage

```
replay(x)
```

Arguments

x	result from `evaluate`

Examples

```
samples <- system.file("tests", "testthat", package = "evaluate")
if (file_test("-d", samples)) {
    replay(evaluate(file(file.path(samples, "order.r"))))
    replay(evaluate(file(file.path(samples, "plot.r"))))
    replay(evaluate(file(file.path(samples, "data.r"))))
}
```

watchout *Watch for changes in output, text and graphical.*

Description

Watch for changes in output, text and graphical.

Usage

```
watchout(debug = FALSE)
```

Arguments

debug	activate debug mode where output will be both printed to screen and captured.

Value

list containing four functions: `get_new`, `pause`, `unpause`, `close`.

Index