Package 'boomer'

July 9, 2024

Title Debugging Tools to Inspect the Intermediate Steps of a Call

Version 0.2.0

Description Provides debugging tools that let you inspect the intermediate results of a call. The output looks as if we explode a call into its parts hence the package name.

License GPL-3

URL https://github.com/moodymudskipper/boomer

BugReports https://github.com/moodymudskipper/boomer/issues

Imports crayon, methods, pryr, rlang, rstudioapi, styler, withr

Suggests flow, knitr, lobstr, magrittr, rmarkdown, shiny, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en

RoxygenNote 7.3.2

NeedsCompilation no

Author Antoine Fabri [aut, cre]

Maintainer Antoine Fabri <antoine.fabri@gmail.com>

Repository CRAN

Date/Publication 2024-07-09 15:50:17 UTC

Contents

boomer-package *boomer: Debugging Tools to Inspect the Intermediate Steps of a Call*

Description

Provides debugging tools that let you inspect the intermediate results of a call. The output looks as if we explode a call into its parts hence the package name.

Details

- `boom()` displays the intermediate results of a call or a code chunk.
- `rig()` creates a copy of a function which will display the intermediate results of all the calls of it body.
- `rig_in_namespace()` rigs a namespaced function in place, so its always verbose even when called by other existing functions. It is especially handy for package development.
- `rigger()` provides a convenient way to rig an anonymous function by using the `rigger(...)` + `function(...) {...}` syntax.
- The addin "Explode a call with 'boom()'" provides a way to `boom()` a call with a keyboard shortcut.

Package options

Several options impact the display of exploded calls :

- `boomer.print`: If the `print` argument is not provided, this option will replace it at run time. Defaults to the `base::print` function.
- `boomer.clock`: If the `clock` argument is not provided, this option will replace it at run time. Defaults to `FALSE`.
- `boomer.print_args`: Whether to print the arguments of rigged functions and their values when they are evaluated. Defaults to `TRUE`.
- `boomer.visible_only`: Whether to hide the output of functions which return invisibly. Defaults to `FALSE`.
- `boomer.ignore`: Vector of function names or named list of functions for which we don't want the result printed (usually because it's redundant). Defaults to `c("~", "{", "(", "<-", "<<-", "=")`.
- `boomer.ignore_args`: Vector of function names or named list of functions for which we don't want the arguments boomed, this might be useful when calling a function that loops too many times.
- `boomer.safe_print`: Whether to replace emoticons by characters compatible with all systems. This is useful for reprexes (see **reprex** package) and for knitted report in case the output of those doesn't look good on your system.
- `boomer.abbreviate`: Whether to show only the function's name rather than the call when it's entered.

Author(s)

Maintainer: Antoine Fabri <antoine.fabri@gmail.com>

See Also

Useful links:

- https://github.com/moodymudskipper/boomer
- Report bugs at https://github.com/moodymudskipper/boomer/issues

---

boom *Print the Output of Intermediate Steps of a Call*

Description

- `boom()` prints the intermediate results of a call or a code chunk.
- `rig()` creates a copy of a function which will display the intermediate results of all the calls of it body.
- `rig_in_namespace()` rigs a namespaced function in place, so its always verbose even when called by other existing functions. It is especially handy for package development.
- `rigger()` provides a convenient way to rig an anonymous function by using the `rigger(...) + function(...) {...}` syntax.

Usage

```
boom(expr, clock = NULL, print = NULL)

rig(fun, clock = NULL, print = NULL)

rigger(clock = NULL, print = NULL)

rig_in_namespace(..., clock = NULL, print = NULL)
```

Arguments

`expr`	call to explode
`clock`	whether to time intermediate steps. Defaults to `getOption("boomer.clock")` evaluated at run time (`FALSE` unless you change it). The execution time of a step doesn't include the execution time of its previously printed sub-steps.
`print`	A function, a formula or a list of functions or formulas, used to modify the way the output is printed. Defaults to `getOption("boomer.print")` evaluated at run time (`base::print` unless you change it)'.
`fun`	function ro `rig()`

`...` Functions to rig in their namespace

If the `print` argument is a function, it will be used to print, or to transform the output before it's printed. Use `invisible` to display nothing, useful possibilities are `str` or `dplyr::glimpse`.

rlang's formula notation is supported, so for instance you can type: `print = ~ dplyr::glimpse(., width = 50)`.

Sometimes you might want to print a specific type of object in a custom way, in this case you can provide a named list, if you provide an unnamed element it will be used as the default, and named elements will define how objects of the given S3 class are printed. For instance `print = list(str, data.frame = tibble::as_tibble)`

Value

`boom()` returns the output of the call. `rig()` returns the modified input function. `rig_in_namespace()` returns `invisible(NULL)` and is called for side effects. `rigger()` returns a list containing the arguments, with the class "rigger" to enable `+.rigger` and `print.rigger`

Examples

```
# explode a simple call
boom(subset(head(mtcars, 2), qsec > 17))

# clock calls and customize how to print output
boom(subset(head(mtcars, 2), qsec > 17), clock = TRUE, print = str)

# print str only for data frames
boom(subset(head(mtcars, 2), qsec > 17), print = list(data.frame = str))

# rig an existing function
rig(ave)(warpbreaks$breaks, warpbreaks$wool)

# rig an anonymous function
fun1 <- rigger() + function(x) x + 1 + 2 # same as rig(function(x) x + 1 + 2))
fun1(1)
fun2 <- rigger(TRUE, typeof) + function(x) x + 1 + 2
fun2(1)
```

---

boom_on *Switch "boom" debugging on and off*

Description

While debugging a function, call `boom_on()` and all subsequent calls will be boomed, call `boom_off()` to return to standard debugging.

Usage

```
boom_on(clock = NULL, print = NULL)

boom_off()
```

Arguments

clock	whether to time intermediate steps. Defaults to `getOption("boomer.clock")` evaluated at run time (`FALSE` unless you change it). The execution time of a step doesn't include the execution time of its previously printed sub-steps.
print	A function, a formula or a list of functions or formulas, used to modify the way the output is printed. Defaults to `getOption("boomer.print")` evaluated at run time (`base::print` unless you change it)'.

Value

Returns `NULL` invisibly, called for side effects.

---

boom_shinyApp *boom the reactive calls of a shiny app*

Description

These works just like `shiny::shinyApp` and `shiny::runApp` and have the exact same parameters, except they create/run a modified app that allows for easier debugging.

Usage

```
boom_shinyApp(
  ui,
  server,
  onStart = NULL,
  options = list(),
  uiPattern = "/",
  enableBookmarking = NULL
)

boom_runApp(
  appDir = getwd(),
  port = getOption("shiny.port"),
  launch.browser = getOption("shiny.launch.browser", interactive()),
  host = getOption("shiny.host", "127.0.0.1"),
  workerId = "",
  quiet = FALSE,
  display.mode = c("auto", "normal", "showcase"),
  test.mode = getOption("shiny.testmode", FALSE)
)
```

Arguments

ui	The UI definition of the app (for example, a call to `fluidPage()` with nested controls). If bookmarking is enabled (see `enableBookmarking`), this must be a single argument function that returns the UI definition.
server	A function with three parameters: `input`, `output`, and `session`. The function is called once for each session ensuring that each app is independent.
onStart	A function that will be called before the app is actually run. This is only needed for `shinyAppObj`, since in the `shinyAppDir` case, a `global.R` file can be used for this purpose.
options	Named options that should be passed to the `runApp` call (these can be any of the following: "port", "launch.browser", "host", "quiet", "display.mode" and "test.mode"). You can also specify `width` and `height` parameters which provide a hint to the embedding environment about the ideal height/width for the app.
uiPattern	A regular expression that will be applied to each `GET` request to determine whether the `ui` should be used to handle the request. Note that the entire request path must match the regular expression in order for the match to be considered successful.
enableBookmarking	Can be one of `"url"`, `"server"`, or `"disable"`. The default value, `NULL`, will respect the setting from any previous calls to `enableBookmarking()`. See `enableBookmarking()` for more information on bookmarking your app.
appDir	Path to directory that contains a Shiny app (i.e. a server.R file and either ui.R or www/index.html)
port	The TCP port that the application should listen on. If the `port` is not specified, and the `shiny.port` option is set (with `options(shiny.port = XX)`), then that port will be used. Otherwise, use a random port between 3000:8000, excluding ports that are blocked by Google Chrome for being considered unsafe: 3659, 4045, 5060, 5061, 6000, 6566, 6665:6669 and 6697. Up to twenty random ports will be tried.
launch.browser	If true, the system's default web browser will be launched automatically after the app is started. Defaults to true in interactive sessions only. The value of this parameter can also be a function to call with the application's URL.
host	The IPv4 address that the application should listen on. Defaults to the `shiny.host` option, if set, or `"127.0.0.1"` if not. See Details.
workerId	Can generally be ignored. Exists to help some editions of Shiny Server Pro route requests to the correct process.
quiet	Should Shiny status messages be shown? Defaults to FALSE.
display.mode	The mode in which to display the application. If set to the value `"showcase"`, shows application code and metadata from a `DESCRIPTION` file in the application directory alongside the application. If set to `"normal"`, displays the application normally. Defaults to `"auto"`, which displays the application in the mode given in its `DESCRIPTION` file, if any.

`test.mode` Should the application be launched in test mode? This is only used for recording or running automated tests. Defaults to the `shiny.testmode` option, or FALSE if the option is not set.

Details

For this function to work properly the main server function should always be assigned to an object (usually you'd name it `server`).

For instance :

- if you have a `server.R` script, make sure to assign your function to `server`
- if you use `shinyServer`, create a `server` function separately and use it in your `shinyServer` call.

It also assumes you follow standard practice in your use of `callModule()` or `moduleServer()`.

Value

See `?shiny::shinyApp` and `?shiny::runApp`

Index