Package 'parsel'

March 17, 2022

Type Package

Title Parallelized Dynamic Web-Scraping Using 'RSelenium'

Version 0.1.1

Description A system to increase the efficiency of dynamic web-scraping with 'RSelenium' by leveraging parallel processing. You provide a function wrapper for your 'RSelenium' scraping routine with a set of inputs, and 'parsel' runs it in several browser instances. Chunked input processing as well as error catching and logging ensures seamless execution and minimal data loss, even when unforeseen 'RSelenium' errors occur.

License MIT + file LICENSE

URL https://github.com/till-tietz/parsel

BugReports https://github.com/till-tietz/parsel/issues

Encoding UTF-8

Imports parallel (>= 3.6.2), RSelenium, lubridate (>= 1.7.9), utils (>= 2.10.1), methods (>= 3.3.1), purrr (>= 0.3.4)

RoxygenNote 7.1.2

Suggests rmarkdown, knitr, testthat (>= 3.0.0), covr (>= 3.5.1)

Config/testthat/edition 3

NeedsCompilation no

Author Till Tietz [cre, aut]

Maintainer Till Tietz <ttietz2014@gmail.com>

Repository CRAN

Date/Publication 2022-03-17 08:20:02 UTC

R topics documented:

parscrape *parallelize execution of RSelenium*

Description

parallelize execution of RSelenium

Usage

```
parscrape(
  scrape_fun,
  scrape_input,
  cores = NULL,
  packages = c("base"),
  browser,
  ports = NULL,
  chunk_size = NULL,
  scrape_tries = 1,
  proxy = NULL,
  extraCapabilities = list()
)
```

Arguments

scrape_fun	a function with input x sending instructions to remDr (remote driver)/ scraping function to be parallelized
scrape_input	a data frame, list, or vector where each element is an input to be passed to scrape_fun
cores	number of cores to run RSelenium instances on. Defaults to available cores - 1.
packages	a character vector with package names of packages used in scrape_fun
browser	a character vector specifying the browser to be used
ports	vector of ports for RSelenium instances. If left at default NULL parscrape will randomly generate ports.
chunk_size	number of scrape_input elements to be processed per round of scrape_fun. parscrape splits scrape_input into chunks and runs scrape_fun in multiple rounds to avoid loosing data due to errors. Defaults to number of cores.
scrape_tries	number of times parscrape will re-try to scrape a chunk when encountering an error
proxy	a proxy setting function that runs before scraping each chunk
extraCapabilities	a list of extraCapabilities options to be passed to rsDriver

Value

a list containing the elements: scraped_results and not_scraped. scraped_results is a list containing the output of scrape_fun. If there are no unscraped input elements then not_scraped is NULL. If there are unscraped elements not_scraped is a data.frame containing the scrape_input id, chunk id and associated error of all unscraped input elements.

Examples

```
## Not run:
input <- c(".central-textlogo__image",".central-textlogo__image")

scrape_fun <- function(x){
 input_i <- x
 remDr$navigate("https://www.wikipedia.org/")
 element <- remDr$findElement(using = "css", input_i)
 element <- element$getElementText()
 return(element)
}

parsel_out <- parscrape(scrape_fun = scrape_fun,
                        scrape_input = input,
                        cores = 2,
                        packages = c("RSelenium"),
                        browser = "firefox",
                        scrape_tries = 1,
                        chunk_size = 2,
                        extraCapabilities = list(
                         "moz:firefoxOptions" = list(args = list('--headless'))
                         )
                        )

## End(Not run)
```

Index