Package 'parsel'

February 22, 2023

Type Package

Title Parallel Dynamic Web-Scraping Using 'RSelenium'

Version 0.3.0

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. You can additionally
build safe scraping functions with minimal coding by utilizing constructor functions that act as wrappers
around 'RSelenium' methods.

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), rlang

RoxygenNote 7.2.2

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 2023-02-22 22:50:02 UTC

R topics documented:

`build_scraper` *generates the scraping function defined by start_scraper and other constructors in your environment*

Description

generates the scraping function defined by start_scraper and other constructors in your environment

Usage

```
build_scraper(prev = NULL)
```

Arguments

`prev`	a placeholder for the output of functions being piped into show(). Defaults to NULL and should not be altered.

Value

a function

Examples

```
## Not run:

start_scraper(args = c("x"), name = "fun") %>>%
go("x") %>>%
build_scraper()



## End(Not run)
```

`click` *wrapper around clickElement() method to generate safe scraping code*

Description

wrapper around clickElement() method to generate safe scraping code

Usage

```
click(using, value, name = NULL, new_page = FALSE, prev = NULL)
```

Arguments

`using`	character string specifying locator scheme to use to search elements. Available schemes: "class name", "css selector", "id", "name", "link text", "partial link text", "tag name", "xpath".
`value`	character string specifying the search target.
`name`	character string specifying the object name the RSelenium "wElement" class object should be saved to.
`new_page`	logical indicating if clickElement() action will result in a change in url.
`prev`	a placeholder for the output of functions being piped into click(). Defaults to NULL and should not be altered.

Value

a character string defining 'RSelenium' clicking instructions that can be pasted into a scraping function.

Examples

```
## Not run:

#navigate to wikipedia, click random article

parsel::go("https://www.wikipedia.org/") %>>%
parsel::click(using = "id", value = "'n-randompage'") %>>%
show()


## End(Not run)
```

get_element *wrapper around getElementText() method to generate safe scraping code*

Description

wrapper around getElementText() method to generate safe scraping code

Usage

```
get_element(using, value, name = NULL, multiple = FALSE, prev = NULL)
```

Arguments

using	character string specifying locator scheme to use to search elements. Available schemes: "class name", "css selector", "id", "name", "link text", "partial link text", "tag name", "xpath".
value	character string specifying the search target.
name	character string specifying the object name the RSelenium "wElement" class object should be saved to. If NULL a name will be generated automatically.
multiple	logical indicating whether multiple elements should be returned. If TRUE the findElements() method will be invoked.
prev	a placeholder for the output of functions being piped into get_element(). Defaults to NULL and should not be altered.

Value

a character string defining 'RSelenium' getElementText() instructions that can be pasted into a scraping function.

Examples

```
## Not run:

#navigate to wikipedia, type "Hello" into the search box,
#press enter, get page header

parsel::go("https://www.wikipedia.org/") %>>%
parsel::type(using = "id",
             value = "'searchInput'",
             name = "searchbox",
             text = c("Hello","\uE007")) %>>%
parsel::get_element(using = "id",
                    value = "'firstHeading'",
                    name = "header") %>>%
            show()

#navigate to wikipedia, type "Hello" into the search box, press enter,
```

```
#get page header, save in external data.frame x.

parsel::go("https://www.wikipedia.org/") %>>%
parsel::type(using = "id",
             value = "'searchInput'",
             name = "searchbox",
             text = c("Hello","\uE007")) %>>%
parsel::get_element(using = "id",
                    value = "'firstHeading'",
                    name = "x[,1]") %>>%
                    show()


## End(Not run)
```

---

go *wrapper around remDr$navigate method to generate safe navigation code*

Description

wrapper around remDr$navigate method to generate safe navigation code

Usage

```
go(url, prev = NULL)
```

Arguments

url	a character string specifying the name of the object holding the url string or the url string the function should navigate to.
prev	a placeholder for the output of functions being piped into go(). Defaults to NULL and should not be altered.

Value

a character string defining 'RSelenium' navigation instructions that can be pasted into a scraping function

Examples

```
## Not run:

go("https://www.wikipedia.org/") %>>%
show()


## End(Not run)
```

goback *wrapper around remDr$goBack method to generate safe backwards navigation code*

Description

wrapper around remDr$goBack method to generate safe backwards navigation code

Usage

```
goback(prev = NULL)
```

Arguments

prev	a placeholder for the output of functions being piped into goback(). Defaults to NULL and should not be altered.

Value

a character string defining 'RSelenium' backwards navigation instructions that can be pasted into a scraping function

Examples

```
## Not run:

goback() %>>%
show()


## End(Not run)
```

goforward *wrapper around remDr$goForward method to generate safe forwards navigation code*

Description

wrapper around remDr$goForward method to generate safe forwards navigation code

Usage

```
goforward(prev = NULL)
```

Arguments

prev	a placeholder for the output of functions being piped into goforward(). Defaults to NULL and should not be altered.

Value

a character string defining 'RSelenium' forward navigation instructions that can be pasted into a scraping function.

Examples

```
## Not run:

goforward() %>>%
show()


## End(Not run)
```

---

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)
```

show *renders the output of the piped functions to the console via cat()*

Description

renders the output of the piped functions to the console via cat()

Usage

```
show(prev = NULL)
```

Arguments

prev	a placeholder for the output of functions being piped into show(). Defaults to NULL and should not be altered.

Value

None (invisible NULL)

Examples

```
## Not run:

go("https://www.wikipedia.org/") %>>%
goback() %>>%
show()


## End(Not run)
```

start_scraper *sets function name and arguments of scraping function*

Description

sets function name and arguments of scraping function

Usage

```
start_scraper(args, name = NULL)
```

Arguments

args	a character vector of function arguments
name	character string specifying the object name of the scraping function. If NULL defaults to 'scraper'

Value

a character string starting a function definition

Examples

```
## Not run:

start_scraper(args = c("x","y"), name = "fun")


## End(Not run)
```

---

type *wrapper around sendKeysToElement() method to generate safe scraping code*

Description

wrapper around sendKeysToElement() method to generate safe scraping code

Usage

```
type(
  using,
  value,
  name = NULL,
  text,
  text_object,
  new_page = FALSE,
  prev = NULL
)
```

Arguments

using	character string specifying locator scheme to use to search elements. Available schemes: "class name", "css selector", "id", "name", "link text", "partial link text", "tag name", "xpath".
value	character string specifying the search target.
name	character string specifying the object name the RSelenium "wElement" class object should be saved to.If NULL a name will be generated automatically.
text	a character vector specifying the text to be typed.
text_object	a character string specifying the name of an external object holding the text to be typed. Note that the remDr$sendKeysToElement method only accepts list inputs.
new_page	logical indicating if sendKeysToElement() action will result in a change in url.
prev	a placeholder for the output of functions being piped into type(). Defaults to NULL and should not be altered.

Value

a character string defining 'RSelenium' typing instructions that can be pasted into a scraping function.

Examples

```
## Not run:

#navigate to wikipedia, type "Hello" into the search box,  press enter

parsel::go("https://www.wikipedia.org/") %>>%
parsel::type(using = "id",
             value = "'searchInput'",
             name = "searchbox",
             text = c("Hello","\uE007")) %>>%
             show()

#navigate to wikipeda, type content stored in external object "x" into search box

parsel::go("https://www.wikipedia.org/") %>>%
parsel::type(using = "id",
             value = "'searchInput'",
             name = "searchbox",
             text_object = "x") %>>%
             show()


## End(Not run)
```

`%>>%` *pipe-like operator that passes the output of lhs to the prev argument of rhs to paste together a scraper function in sequence.*

Description

pipe-like operator that passes the output of lhs to the prev argument of rhs to paste together a scraper function in sequence.

Usage

```
lhs %>>% rhs
```

Arguments

`lhs`	a parsel constructor function call
`rhs`	a parsel constructor function call that should accept lhs as its prev argument

Value

the output of rhs evaluated with lhs as the prev argument

Examples

```
## Not run:

#paste together the go and goback output in sequence
go("https://www.wikipedia.org/") %>>%
goback()


## End(Not run)
```

Index