Package 'webdriver'

April 11, 2018

Title 'WebDriver' Client for 'PhantomJS'

Version 1.0.5

Author Ariya Hidayat, Gábor Csárdi, Gabor Torok, Ivan De Marino, Robert Gieseke, Steven Masala, Winston Chang, Zack Weinberg

Maintainer Gábor Csárdi <csardi.gabor@gmail.com>

Description A client for the 'WebDriver' 'API'. It allows driving a (probably headless) web browser, and can be used to test web applications, including 'Shiny' apps. In theory it works with any 'WebDriver' implementation, but it was only tested with 'PhantomJS'.

License MIT + file LICENSE

LazyData true

URL https://github.com/rstudio/webdriver

BugReports https://github.com/rstudio/webdriver/issues

RoxygenNote 6.0.1

Imports callr (>= 2.0.0), base64enc, curl (>= 2.0), debugme, httr, jsonlite, R6, showimage, utils, withr

Suggests covr, pingr, rprojroot, servr, testthat

Encoding UTF-8

SystemRequirements PhantomJS (http://phantomjs.org/)

NeedsCompilation no

Repository CRAN

Date/Publication 2018-04-11 03:48:44 UTC

R topics documented:

Element   *HTML element*

Description

HTML element

Usage

```
e <- s$findElement(css = NULL, linkText = NULL,
    partialLinkText = NULL, xpath = NULL)

e$findElement(css = NULL, linkText = NULL,
    partialLinkText = NULL, xpath = NULL)
e$findElements(css = NULL, linkText = NULL,
    partialLinkText = NULL, xpath = NULL)

e$isSelected()
e$getValue()
e$setValue(value)
e$getAttribute(name)
e$getClass()
e$getCssValue(name)
e$getText()
e$getName()
e$getData(name)
e$getRect()
e$isEnabled()
e$click()
e$clear()
e$sendKeys(...)
e$moveMouseTo(xoffset = NULL, yoffset = NULL)

e$executeScript(script, ...)
e$executeScriptAsync(script, ...)
```

Arguments

e An Element object.

s A [Session](#) object.

css Css selector to find an HTML element.

linkText Find <a> HTML elements based on their innerText.

partialLinkText Find <a> HTML elements based on their innerText. It uses partial matching.

xpath Find HTML elements using XPath expressions.

name String scalar, named of attribute, property or css key. For getData, the key of the data attribute.

xoffset Horizontal offset for mouse movement, relative to the position of the element. If at least of of xoffset and yoffset is NULL, then they are ignored.

yoffset Vertical offset for mouse movement, relative to the position of the element. If at least of of xoffset and yoffset is NULL, then they are ignored.

value Value to set, a character string.

... For sendKeys the keys to send, see key. For executeScript and executeScriptAsync argument to supply to the script.

Details

To create Element objects, you need to use the findElement (or findElement) method of a Session object.

e$findElement() finds the *next* HTML element from the current one. You need to specify one of the css, linkText, partialLinkText and xpath arguments. It returns a new Element object.

e$findElements() finds all matching HTML elements starting from the current element. You need to specify one of the css, linkText, partialLinkText and xpath arguments. It returns a list of newly created Element objects.

e$isSelected() returns TRUE is the element is currently selected, and FALSE otherwise.

e$getValue() returns the value of an input element, it is a shorthand for e$getAttribute("value").

e$setValue() sets the value of an input element, it is essentially equivalent to sending keys via e$sendKeys().

e$getAttribute() queries an arbitrary HTML attribute. It is does not exist, NULL is returned.

e$getClass() uses e$getAttribute to parse the 'class' attribute into a character vector.

e$getCssValue() queries a CSS property of an element.

e$getText() returns the innerText on an element.

e$getName() returns the tag name of an element.

e$getData() is a shorthand for querying data-* attributes.

e$getRect() returns the 'rectangle' of an element. It is named list with components x, y, height and width.

e$isEnabled() returns TRUE if the element is enabled, FALSE otherwise.

e$click() scrolls the element into view, and clicks the in-view centre point of it.

e$clear() scrolls the element into view, and then attempts to clear its value, checkedness or text content.

e$sendKeys() scrolls the form control element into view, and sends the provided keys to it. See key for a list of special keys that can be sent.

e$uploadFile() uploads a file to a <input type="file"> element. The filename argument can contain a single filename, or multiple filenames, for file inputs that can take multiple files.

e$moveMouseTo() moves the mouse cursor to the element, with the specified offsets. If one or both offsets are NULL, then it places the cursor on the center of the element. If the element is not on the screen, then is scrolls it into the screen first.

e$executeScript() and e$executeScriptAsync() call the method of the same name on the
Session object. The first argument of the script (arguments[0]) will always hold the element
object itself.

---

install_phantomjs   *Install PhantomJS*

Description

Download the zip package, unzip it, and copy the executable to a system directory in which **web-
driver** can look for the PhantomJS executable.

Usage

```
install_phantomjs(version = "2.1.1",
  baseURL = "https://github.com/wch/webshot/releases/download/v0.3.1/")
```

Arguments

version   The version number of PhantomJS.

baseURL   The base URL for the location of PhantomJS binaries for download. If the
                 default download site is unavailable, you may specify an alternative mirror, such
                 as "https://bitbucket.org/ariya/phantomjs/downloads/".

Details

This function was designed primarily to help Windows users since it is cumbersome to modify
the PATH variable. Mac OS X users may install PhantomJS via Homebrew. If you download the
package from the PhantomJS website instead, please make sure the executable can be found via the
PATH variable.

On Windows, the directory specified by the environment variable APPDATA is used to store 'phantomjs.exe'.
On OS X, the directory '~/Library/Application Support' is used. On other platforms (such as
Linux), the directory '~/bin' is used. If these directories are not writable, the directory 'PhantomJS'
under the installation directory of the **webdriver** package will be tried. If this directory still fails,
you will have to install PhantomJS by yourself.

Value

NULL (the executable is written to a system directory).

---

key   *Special keys, so that we can refer to them with an easier syntax*

Description

Special keys, so that we can refer to them with an easier syntax

Usage

```
key
```

Format

An object of class list of length 51.

Examples

```
## Not run:
el$sendKeys("xyz")
el$sendKeys("x", "y", "z")
el$sendKeys("username", key$enter, "password", key$enter)

## Sending CTRL+A
el$sendKeys(key$control, "a")

## Note that modifier keys (control, alt, shift, meta) are sticky,
## they remain in effect in the rest of the sendKeys() call. E.g.
## this sends CTRL+X and CTRL+S
el$sendKeys(key$control, "x", "s")

## You will need multiple calls to release control and send CTRL+X S
el$sendKeys(key$control, "x")
el$sendKeys("s")

## End(Not run)
```

---

run_phantomjs   *Start up phantomjs on localhost, and a random port*

Description

Throws and error if phantom cannot be found, or cannot be started. It works with a timeout of five seconds.

Usage

```
run_phantomjs(debugLevel = c("INFO", "ERROR", "WARN", "DEBUG"),
  timeout = 5000)
```

Arguments

debugLevel   Phantom.js debug level, possible values: "INFO", "ERROR", "WARN", "DEBUG".

timeout   How long to wait (in milliseconds) for the webdriver connection to be established to the phantomjs process.

Value

A list of `process`, the `callr::process` object, and `port`, the local port where phantom is running.

Session   *WebDriver session*

Description

Drive a headless phantom.js browser via the WebDriver protocol. It needs phantom.js running in WebDriver mode.

Usage

```
s <- Session$new(host = "127.0.0.1", port = 8910)

s$delete()
s$status()

s$go(url)
s$getUrl()
s$goBack()
s$goForward()
s$refresh()
s$getTitle()
s$getSource()
s$takeScreenshot(file = NULL)

s$findElement(css = NULL, linkText = NULL,
    partialLinkText = NULL, xpath = NULL)
s$findElements(css = NULL, linkText = NULL,
    partialLinkText = NULL, xpath = NULL)

s$executeScript(script, ...)
s$executeScriptAsync(script, ...)

s$setTimeout(script = NULL, pageLoad = NULL, implicit = NULL)
```

```
s$moveMouseTo(xoffset = 0, yoffset = 0)
s$click(button = c("left", "middle", "right"))
s$doubleClick(button = c("left", "middle", "right"))
s$mouseButtonDown(button = c("left", "middle", "right"))
s$mouseButtonUp(button = c("left", "middle", "right"))

s$readLog(type = c("browser", "har"))
s$getLogTypes()

s$waitFor(expr, checkInterval = 100, timeout = 3000)
```

Arguments

s   A `Session` object.

host   Host name of phantom.js.

port   Port of phantom.js.

url   URL to nagivate to.

file   File name to save the screenshot to. If `NULL`, then it will be shown on the R graphics device.

css   Css selector to find an HTML element.

linkText   Find HTML elements based on their `innerText`.

partialLinkText   Find HTML elements based on their `innerText`. It uses partial matching.

xpath   Find HTML elements using XPath expressions.

script   For `executeScript` and `executeScriptAsync`. JavaScript code to execute. It will be placed in the body of a function.

...   Arguments to the script, they will be put in a list called arguments. [Element](#) objects are automatically transformed to DOM element in JavaScript.

script   For `setTimeout`. Script execution timeout, in milliseconds. More below.

pageLoad   Page load timeout, in milliseconds. More below.

implicit   Implicit wait before calls that find elements, in milliseconds. More below.

xoffset   Horizontal offset for mouse movement, relative to the current position.

yoffset   Vertical offset for mouse movement, relative to the current position.

button   Mouse button. Either one of `"left"`, `"middle"`, `"right"`, or an integer between 1 and 3.

type   Log type, a character scalar.

expr   A string scalar containing JavaScript code that evaluates to the condition to wait for.

checkInterval   How often to check for the condition, in milliseconds.

timeout   Timeout for the condition, in milliseconds.

Details

`Session$new()` creates a new WebDriver session.

`s$delete()` deletes a WebDriver session.

`s$status()` returns a status message from the server. It is a named list, and contains version numbers and capabilities.

`s$go()` navigates to the supplied URL.

`s$getUrl()` returns the current URL.

`s$goBack()` is like the web browser's back button. It goes back to the previous page.

`s$goForward()` is like the web browser's forward button.

`s$refresh()` is like the web browser's refresh button.

`s$getTitle()` returns the title of the current page.

`s$getSource()` returns the complete HTML source of a page, in a character scalar.

`s$takeScreenshot()` takes a screenshot of the current page. You can save it to a PNG file with the `file` argument, or show it on the graphics device (if `file` is NULL).

`s$findElement()` finds a HTML element using a CSS selector, XPath expression, or the `innerHTML` of the element. If multiple elements match, then the first one is returned. The return value is an [Element](#) object.

`s$findElements()` finds HTML elements using a CSS selector, XPath expression, or the `innerHTML` of the element. All matching elements are returned in a list of [Element](#) objects.

`s$executeScript()` executes JavaScript code. It places the code in the body of a function, and then calls the function with the additional arguments. These can be accessed from the function via the `arguments` array. Returned DOM elements are automatically converted to [Element](#) objects, even if they are inside a list (or list of list, etc.).

`s$executeScriptAsync()` is similar, for asynchronous execution. It place the script in a body of a function, and then calls the function with the additional arguments and a callback function as the last argument. The script must call this callback function when it finishes its work. The first argument passed to the callback function is returned. Returned DOM elements are automatically converted to [Element](#) objects, even if they are inside a list (or list of list, etc.).

`s$setTimeout()` sets various timeouts. The 'script' timeout specifies a time to wait for scripts to run. The sQuotepage load timeout specifies a time to wait for the page loading to complete. The 'implicit' specifies a time to wait for the implicit element location strategy when locating elements. Their defaults are different in the standard and in Phantom.js. In Phantom.js the 'script' and 'page load' timeouts are set to infinity, and the 'implicit' waiting time is 200ms.

`s$moveMouseTo()` moves the mouse cursor by the specified offsets.

`s$click()` clicks the mouse at its current position, using the specified button.

`s$doubleClick()` emulates a double click with the specified mouse button.

`s$button_down()` emulates pressing the specified mouse button down (and keeping it down).

`s$button_up()` emulates releasing the specified mouse button.

`s$getLogTypes()` returns the log types supported by the server, in a character vector.

`s$readLog()` returns the log messages since the last `readLog` call, in a data frame with columns `timestamp`, `level` and `message`.

s$waitFor() waits until a JavaScript expression evaluates to `true`, or a timeout happens. It returns `TRUE` is the expression evaluated to `true`, possible after some waiting. If the expression has a syntax error or a runtime error happens, it returns `NA`.

See Also

The WebDriver standard at <https://w3c.github.io/webdriver/webdriver-spec.html>.

---

webdriver	*'WebDriver' Client for 'PhantomJS'*

Description

A client for the 'WebDriver' 'API'. It allows driving a (probably headless) web browser, and can be used to test web applications, including 'Shiny' apps. In theory it works with any 'WebDriver' implementation, but it was only tested with 'PhantomJS'.

Window	*A browser window*

Description

A browser window

Usage

```
w <- s$getWindow()
wlist <- s$getAllWindows()

w$close()
w$isActive()
w$switchTo()
w$maximize()
w$getSize()
w$setSize(width, height)
w$getPosition()
w$setPosition(x, y)
```

Arguments

s A [Session](#) object.

w A Window object.

wlist A list of Window objects.

width Integer scalar, requested width of the window.

height Integer scalar, requested height of the window.

x Integer scalar, requested horizontal window position.

y Integer scalar, requested vertical window position.

Details

The `getWindow` method of a [Session](#) object returns the current browser window as a `Window` object. The `getAllWindows` method returns a list of window objects, all browser windows.

`w$close()` closes the window.

`w$isActive()` returns `TRUE` if the window is active, `FALSE` otherwise.

`w$switchTo` makes the window active.

`w$maximize` maximizes the window. Currently it sets it to a fixed size.

`w$getSize` returns the size of the window, in a list with elementh `width` and `height`, both integers.

`w$setSize` sets the size of the window.

`w$getPosition` returns the position of the window on the screen. Phantom.js being headless, it always returns `list(x = 0, y = 0)`, and it is included to have a complete impelementation of the WebDriver standard.

`w$setPosition(x, y)` sets the position of the window on the screen. Phantom.js being headless, it has no effect, and it is included to have a complete implementation of the WebDriver standard.

Index