Common Lisp Selenium Webdriver

A.U. Thor <bug-sample@gnu.org>

Table of Contents

1 Introduction

CL Selenium WebDriver is a binding library to the Selenium 2.0

This software is in development. The APIs will be likely to change.

2 Usage

```lisp
;; see examples/*.lisp and t/*.lisp
(in-package :cl-user)

(eval-when (:compile-toplevel :load-toplevel :execute)
  (ql:quickload :cl-selenium))

(defpackage go-test
  (:use :cl :cl-selenium))

(in-package :go-test)

(defparameter *code* "
package main
import \"fmt\"

func main() {
    fmt.Print(\"Hello WebDriver!\")
}")

(with-session ()
  (setf (url) "http://play.golang.org/?simple=1")
  (let ((elem (find-element "#code" :by :css-selector)))
    (element-clear elem)
    (element-send-keys elem *code*))
  (let ((btn (find-element "#run")))
    (element-click btn))

  (loop
     with div = (find-element "#output")
     for ouput = (element-text div)
     while (equal ouput "Waiting for remote server...")
     do (sleep 0.1)
     finally (print ouput)))
```

3 Installation

```
git clone https://github.com/TatriX/cl-selenium-webdriver ~/quicklisp/local-
projects/
(ql:quickload :cl-selenium)
```

You need a running instance of selenium-server-standalone.

[Download](http://www.seleniumhq.org/download/) it and run:

```
curl -L0 https://goo.gl/SP94ZB -o selenium-server-standalone.jar
java -jar selenium-server-standalone.jar
```

4 Utils

There is a `:cl-selenium-utils` package which should reduce boilerplate. For example:

```
(defpackage my-test
  (:use :cl :cl-selenium)
  (:import-from :cl-selenium-utils
                :send-keys
                :click
                :wait-for
                :classlist))

(in-package :my-test)

(with-session ()
  (setf (url) "http://google.com")
  (send-keys "cl-selenium-webdriver")
  (click "[name=btnK]")
  (wait-for "#resultStats"))
```

4.1 Interactive session

You can just start the session and control it from your repl:

```
(in-package :my-test)

(start-interactive-session)

(setf (url) "http://google.com")
(send-keys "cl-selenium-webdriver")
(send-keys (key :enter))
(classlist "#slim_appbar") ; prints ("ab_tnav_wrp")

(stop-interactive-session)
```

4.2 Utils API conventions

If utility function needs an element to work on it defaults to '(active-element)'.

```
(click) ; click on the current active element.
```

You can also pass a css selector as a last parameter.

```
(print (id "#submit")) ; print id the of matched element
```

```
(assert (= (first (classlist "div")) "first-div-ever"))
```

To change default element you can:

```
(setf cl-selenium-utils:*default-element-func* (lambda () (find-element "input[type=su
```

4.3 Waiting for the reaction

Often you need to wait for some action to be done. For example if you do a `(click)` on the button to load search results, you need to wait them to load.

```
(wait-for ".search-result" :timeout 10) ; wait 10 seconds
```

Timeout defaults to 30 seconds. You can globally change it:

```
(setf cl-selenium-utils:*timeout* 3)
```

4.4 Running tests

REPL

```
(ql:quickload '(:cl-selenium :prove))
(setf prove:*enable-colors* nil)
(prove:run :cl-selenium-test)
```

Shell

```
sh
./test.sh
```

5 API

5.1 CL-SELENIUM package

CL-SELENIUM   [PACKAGE]

This package exports functions for working with Selenium WebDriver.

For documentation see: - https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol█
- https://www.w3.org/TR/webdriver1.

External definitions

Functions

CLOSE-CURRENT-WINDOW (**&key** *(session *session*)*)   [CL-SELENIUM]

Close the current window.

COOKIE (**&key** *(session *session*)*)   [CL-SELENIUM]

Retrieve all cookies visible to the current page.

See: https://www.w3.org/TR/webdriver1/#get-all-cookies.
See: https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#sessionsessionidcookie.█

URL (**&key** *(session *session*)*)   [CL-SELENIUM]

Get the current url in *session*.
See: https://www.w3.org/TR/webdriver1/#dfn-get-current-url.

ELEMENT-SEND-KEYS (*element keys* **&key** *(session *session*)*)   [CL-SELENIUM]

The *Element* Send *Keys* command scrolls into view the form control *element* and
then sends the provided *keys* to the *element*. In case the *element* is not keyboard-
interactable, an *element* not interactable error is returned.

See: https://www.w3.org/TR/webdriver1/#element-send-keys.

ELEMENT-CLICK (*element* **&key** *(session *session*)*)   [CL-SELENIUM]

The *Element* Click command scrolls into view the *element* if it is not already pointer-
interactable, and clicks its in-view center point.

If the *element*'s center point is obscured by another *element*, an *element* click in-
tercepted error is returned. If the *element* is outside the viewport, an *element* not
interactable error is returned.

See: https://www.w3.org/TR/webdriver1/#element-click.

LOG-TYPES (**&key** *(session *session*)*)   [CL-SELENIUM]

Return the types of logs supported by the WebDriver.

- browser: Javascript console logs from the browser.

- client: Logs from the client side implementation of the WebDriver protocol (e.g. the Java bindings).
- driver: Logs from the internals of the driver (e.g. FirefoxDriver internals).
- performance: Logs relating to the performance characteristics of the page under test (e.g. resource load timings).
- server: Logs from within the selenium server.

See: https://github.com/SeleniumHQ/selenium/wiki/Logging.

FIND-ELEMENTS (*value* **&key** *(by :css-selector) (session*   [CL-SELENIUM]
        session))
        Find elements that match *VALUE* using location strategy in *BY*.
        See See [CL-SELENIUM:FIND-ELEMENT function], page 8.
        See https://www.w3.org/TR/webdriver1/#find-elements.

MOUSE-MOVE-TO (*x y* **&key** *element (session *session*))*   [CL-SELENIUM]
        Move the mouse by an offset of the specified *element*. If no *element* is specified, the move is relative to the current mouse cursor. If an *element* is provided but no offset, the mouse will be moved to the center of the *element*. If the *element* is not visible, it will be scrolled into view.

        See: https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#sessionsessionidmoveto█

ELEMENT-ATTRIBUTE (*element name* **&key** *(session *session*))*   [CL-SELENIUM]
        Return the *ELEMENT*'s attribute named *NAME*.

SWITCH-TO-FRAME (*id* **&key** *(session *session*))*   [CL-SELENIUM]
        Change focus to another frame on the page. If the frame *id* is null, the server should switch to the page's default content.

        In the context of a web browser, a frame is a part of a web page or browser window which displays content independent of its container, with the ability to load content independently.

        See: https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#sessionsessionidframe.█

        See: https://en.wikipedia.org/wiki/Frame_(World_Wide_Web).

EXECUTE-SCRIPT (*script args* **&key** *(session *session*))*   [CL-SELENIUM]
        Inject a snippet of JavaScript into the page for execution in the context of the currently selected frame. The executed *script* is assumed to be synchronous and the result of evaluating the *script* is returned to the client.

        The *script* argument defines the *script* to execute in the form of a function body. The value returned by that function will be returned to the client. The function will be invoked with the provided *args* array and the values may be accessed via the arguments object in the order specified.

        Arguments may be any JSON-primitive, array, or JSON object. JSON objects

that define a WebElement reference will be converted to the corresponding DOM element. Likewise, any WebElements in the *script* result will be returned to the client as WebElement JSON objects.

See: https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#sessionsessionidexecute.■

START-INTERACTIVE-SESSION (**&rest** *capabilities*)   [CL-SELENIUM]
Start an interactive session. Use this to interact with Selenium driver from a REPL.

SCREENSHOT (**&key** *(session *session*)*)   [CL-SELENIUM]
Screenshots are a mechanism for providing additional visual diagnostic information. They work by dumping a snapshot of the initial viewport's framebuffer as a lossless PNG image. It is returned to the local end as a Base64 encoded string.
See: https://www.w3.org/TR/webdriver2/#screen-capture.

FIND-ELEMENT (*value* **&key** *(by :css-selector) (session*   [CL-SELENIUM]
          session))
The Find Element command is used to find an element in the current browsing context that can be used as the web element context for future element-centric commands.

For example, consider this pseudo code which retrieves an element with the #toremove ID and uses this as the argument for a script it injects to remove it from the HTML document:

let body See ⟨undefined⟩ [COMMON-LISP:= function], page ⟨undefined⟩ *session*.find.css("#toremove");
session.execute("arguments[0].remove()", [body]);

The *BY* parameter represents the element location strategy.

It can be one of:
- *:id* : Finds element *by* id.
- *:class-name* : Finds element *by* class name.
- *:css-selector* : Returns element that matches css selector.
- *:link-text* : Returns element that matches <a> element text.
- *:partial-link-text:* Returns element that matches <a> element text partially.
- *:tag-name:* Returns element that matches tag name.
- *:xpath:* Returns element that matches the XPath expression.

If result is empty, a See ⟨undefined⟩ [CL-SELENIUM:HANDLE-FIND-ERROR function], page ⟨undefined⟩ is signaled.

See: https://www.w3.org/TR/webdriver1/#dfn-find-element.

MOUSE-CLICK (*button* **&key** *(session *session*)*)   [CL-SELENIUM]
Click any mouse *button* (at the coordinates set by the last moveto command). Note that calling this command after calling buttondown and before calling *button* up (or

any out-of-order interactions sequence) will yield undefined behaviour).

See: https://github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol#sessionsessionidclick▮

USE-SESSION (*session*)   [CL-SELENIUM]
>   Make *SESSION* the current *session*.

PAGE-TITLE (**&key** (*session *session**))   [CL-SELENIUM]
>   This command returns the document title of the current top-level browsing context,
>   equivalent to calling document.title.
>   See: https://www.w3.org/TR/webdriver2/#get-title.

LOGS (*type* **&key** (*session *session**))   [CL-SELENIUM]
>   Return the logs of a particular *TYPE*.
>   See: See [CL-SELENIUM:LOG-TYPES function], page 6.

ELEMENT-LOCATION (*element* **&key** (*session *session**))   [CL-SELENIUM]
>   Return the *ELEMENT*'s location.

ELEMENT-TEXT (*element* **&key** (*session *session**))   [CL-SELENIUM]
>   The Get *Element* Text command intends to return an *element*'s text "as rendered".
>   An *element*'s rendered text is also used for locating a elements by their link text and
>   partial link text.
>
>   See: https://www.w3.org/TR/webdriver1/#get-element-text.

ACTIVE-ELEMENT (**&key** (*session *session**))   [CL-SELENIUM]
>   Return the active element of the current browsing context's document.
>   The active element is the Element within the DOM that currently has focus.
>   If there's no active element, an error is signaled.
>
>   See: https://www.w3.org/TR/webdriver2/#get-active-element.
>   See: https://developer.mozilla.org/en-US/docs/Web/API/Document/activeElement.▮

MAKE-COOKIE (*name value* **&key** *path domain secure expiry*)   [CL-SELENIUM]

STOP-INTERACTIVE-SESSION *nil*   [CL-SELENIUM]
>   Stop an interactive session.

ELEMENT-TAGNAME (*element* **&key** (*session *session**))   [CL-SELENIUM]
>   Return the *ELEMENT*'s tag name.

REFRESH (**&key** (*session *session**))   [CL-SELENIUM]
>   Refresh the current page.

ELEMENT-DISPLAYED (*element* **&key** (*session *session**))   [CL-SELENIUM]
>   Returns if *ELEMENT* is visible.
>
>   See: https://www.w3.org/TR/webdriver1/#element-displayedness.

DELETE-SESSION (*session*)   [CL-SELENIUM]
>   Delete the WebDriver *SESSION*.

MAKE-SESSION (**&key** *(browser-name :chrome) browser-version*   [CL-SELENIUM]
          platform-name platform-version accept-ssl-certs additional-capabilities)
    Creates a new WebDriver session with the endpoint node. If the creation fails, a
session not created error is returned.

    See: https://www.w3.org/TR/webdriver1/#new-session.
    See: https://www.w3.org/TR/webdriver1/#capabilities.

KEY (*key*)   [CL-SELENIUM]

ELEMENT-CLEAR (*element* **&key** *(session *session*)*)   [CL-SELENIUM]
    Clear the contents of *ELEMENT* (for example, a form field *element*).

    See: https://www.w3.org/TR/webdriver1/#dfn-element-clear.

BACK (**&key** *(session *session*)*)   [CL-SELENIUM]
    This command causes the browser to traverse one step backward in the joint *session*
history of the current top-level browsing context. This is equivalent to pressing the
back button in the browser chrome or invoking window.history.back.
    See: https://www.w3.org/TR/webdriver1/#dfn-back.

Classes

COOKIE   [CL-SELENIUM]

NO-SUCH-ELEMENT-ERROR   [CL-SELENIUM]
    Error signaled when no such element is found.

5.2 CL-SELENIUM-UTILS package

CL-SELENIUM-UTILS   [PACKAGE]
    Package with the purpose of reducing boilerplate.

External definitions

Variables

TIMEOUT   [CL-SELENIUM-UTILS]
    Default timeout value to use in selenium-utils functions.

DEFAULT-ELEMENT-FUNC   [CL-SELENIUM-UTILS]
    Function used to get the 'default element' by selenium-utils functions.
    It is See [CL-SELENIUM:ACTIVE-ELEMENT function], page 9 function by default.

Functions

ID (**&optional** *selector*)   [CL-SELENIUM-UTILS]
    Get active element id.

GET-COOKIE (*cookie name*)   [CL-SELENIUM-UTILS]
      Get value of *COOKIE* at *NAME*.

FIND-ELEM (*selector* **&key** *(by :css-selector)*)   [CL-SELENIUM-UTILS]
      Find element *by SELECTOR*. Returns NIL if the element is not found.

WAIT-FOR (*selector* **&key** *(timeout *timeout*)*)   [CL-SELENIUM-UTILS]
      Wait for an element that matches *SELECTOR* to appear on the screen.
      TIMEOUT indicates how much time to wait (default is **TIMEOUT**).

CLASSNAME (**&optional** *selector*)   [CL-SELENIUM-UTILS]
      Get active element classname.

TEXT (**&optional** *selector*)   [CL-SELENIUM-UTILS]
      Get active element's text.

SEND-KEY (*key* **&optional** *selector*)   [CL-SELENIUM-UTILS]
      Send a *key* to active element.

CLASSLIST (**&optional** *selector*)   [CL-SELENIUM-UTILS]
      Get active element class list.

ATTR (*name* **&optional** *selector*)   [CL-SELENIUM-UTILS]
      Get acttive element attribute.

SEND-KEYS (*keys* **&optional** *selector*)   [CL-SELENIUM-UTILS]
      Send *keys* to active element.

ELEM (**&optional** *selector*)   [CL-SELENIUM-UTILS]
      If *SELECTOR* is given, wait for an element that matches the *selector* to appear.
      Otherwise, call See [CL-SELENIUM-UTILS:*DEFAULT-ELEMENT-FUNC* vari-
      able], page 10 (the active element is returned by default).

CLICK (**&optional** *selector*)   [CL-SELENIUM-UTILS]
      Click on active element.

6 Index

(Index is nonexistent)

K

L

M

P

R

S

T

U

W