Automation with Selenium

What is Selenium?

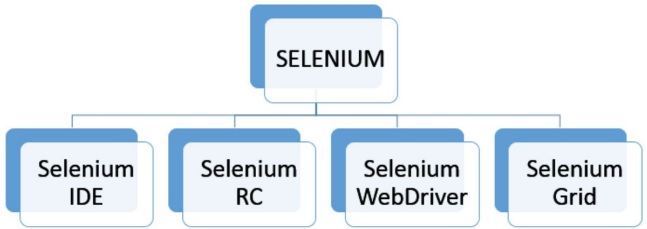
Selenium automates browsers. That's it! What you do with that power is entirely up to you. Primarily, it is for automating web applications for testing purposes, but is certainly not limited to just that.

- Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms.

- The languages supported are Python, Java, C#, Ruby, Javascript (Node).

The major four components of Selenium are:

1. Selenium IDE
2. Selenium RC
3. WebDriver
4. Selenium Grid



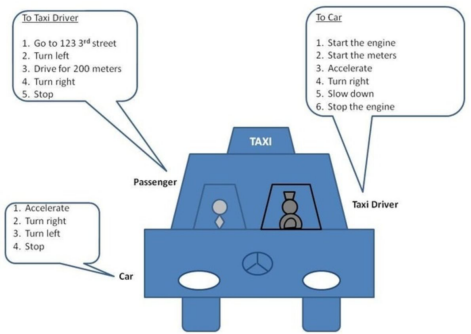
How Selenium webdriver works?

There are 2 ways of explaining how Selenium WebDriver works:

Simple way - Selenium WebDriver drives a browser the same way a taxi driver drives a cab.

In taxi driving, there are typically 3 actors:

- **the client:** he tells the taxi driver where he wants to go and how to get there
- **the taxi driver:** he executes the client's requests; the taxi driver sends his own requests to the car
- **the car:** the car executes the taxi driver's requests

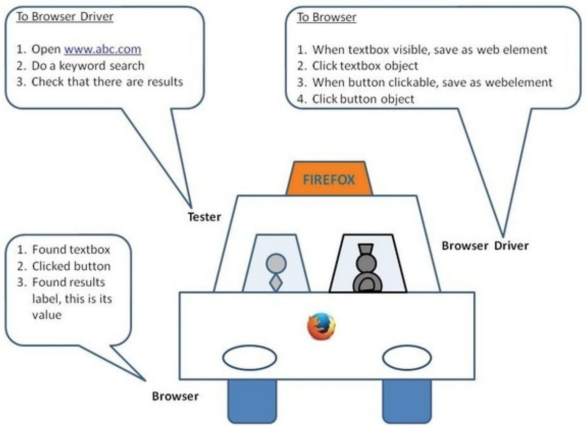


The client gets to the destination through dialogues that happen between the client - taxi driver and taxi driver - car.

How Selenium webdriver works?

In test automation with Selenium WebDriver (and other tools), there are 3 actors as well:

- **test engineer that writes the automation code**: the automation code sends requests to the browser driver component
- **the browser driver component**: it executes the test engineer requests; it sends its own request to the browser
- **the browser**: it executes the browser driver requests



How Selenium webdriver works?

Like most technical explanations, there are no photos either :(

When the automation script is executed, the following steps happen:

- **for each Selenium command, a HTTP request is created and sent to the browser driver**
- the browser driver **uses a HTTP server for getting the HTTP requests**
- **the HTTP server** determines the **steps needed for implementing the Selenium command**
- the implementation steps are executed on the browser
- the execution status is sent back to the HTTP server
- the **HTTP server sends the status back to the automation script**

Setting up Selenium

Installing Selenium

```
pip install -U selenium
```

Browser driver executables

Chrome: https://sites.google.com/a/chromium.org/chromedriver/downloads
Edge: https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/
Firefox: https://github.com/mozilla/geckodriver/releases
Safari: https://webkit.org/blog/6900/webdriver-support-in-safari-10/

Refer to - https://www.seleniumhq.org/download/
https://seleniumhq.github.io/selenium/docs/api/py/index.html

Selenium Source code - https://github.com/SeleniumHQ/selenium

Opening a browser using Selenium

```
from selenium import webdriver

browser = webdriver.Chrome("Enter your browser driver path")

browser.get('http://www.yahoo.com')
browser.maximize_window()
assert 'Yahoo' in browser.title

browser.close()
```

Locators

There are various strategies to locate elements in a page. You can use the most appropriate one for your case. Below are the ones which are used in Selenium.

1. ID
2. Name
3. Class name
4. Link text
5. Partial link text
6. Tag name
7. CSS selectors
8. Xpath

Locating Element(s)

Single Element	Multiple Elements
find_element_by_id	-
find_element_by_name	*find_elements_by_name*
find_element_by_xpath	*find_elements_by_xpath*
find_element_by_link_text	*find_elements_by_link_text*
find_element_by_partial_link_text	*find_elements_by_partial_link_text*
find_element_by_tag_name	*find_elements_by_tag_name*
find_element_by_class_name	*find_elements_by_class_name*
find_element_by_css_selector	*find_elements_by_css_selector*

Page navigation and interactions

Page navigation

- driver.get(“URL”)
- driver.forward()
- driver.back()
- driver.refresh()

Page interactions

- element.click()
- element.send_keys(“Text”)
- element.clear()
- element.submit()
- Select
- ActionChains

Synchronization

- **Implicit Waits**

  driver.implicitly_wait(“seconds”)

- **Explicit Waits**

  element = WebDriverWait(driver, “seconds”).until(EC.”Condition to wait for”((By.”locator type”, “locator”)))

- **Fluent Waits**

  element = WebDriverWait(driver, “seconds”, “poll frequency ”, “ignored_exceptions”).until(EC.”Condition to wait for”((By.”locator type”, “locator”)))

Javascript Executor

2 ways to work with Javascript Executor

- **Executing Javascript at document root level**
- **Executing Javascript at element level**

Handling Windows and Frames

Handling Windows

driver.switch_to_window(“windowName”)

Handling Frames

driver.switch_to_frame(“frameName”)

Handling PopUps

- **Javascript Pop Ups**
- **Hidden division Pop Ups**
- **File Download Pop Ups**
- **File Upload Pop Ups**

Screenshot and Logging

Screenshots

- driver.save_screenshot(“filename.png”)

Logging

- Debug
- Info
- Warning
- Error
- Critical