Scraping with Selenium

by Dominique Theodore

Overview

1. What is Selenium
2. Selenium vs. Scrapy
3. Download and installation
4. First steps
5. Finding elements
6. Using locators
7. Scraping dynamic pages
8. Infinite scrolling pages

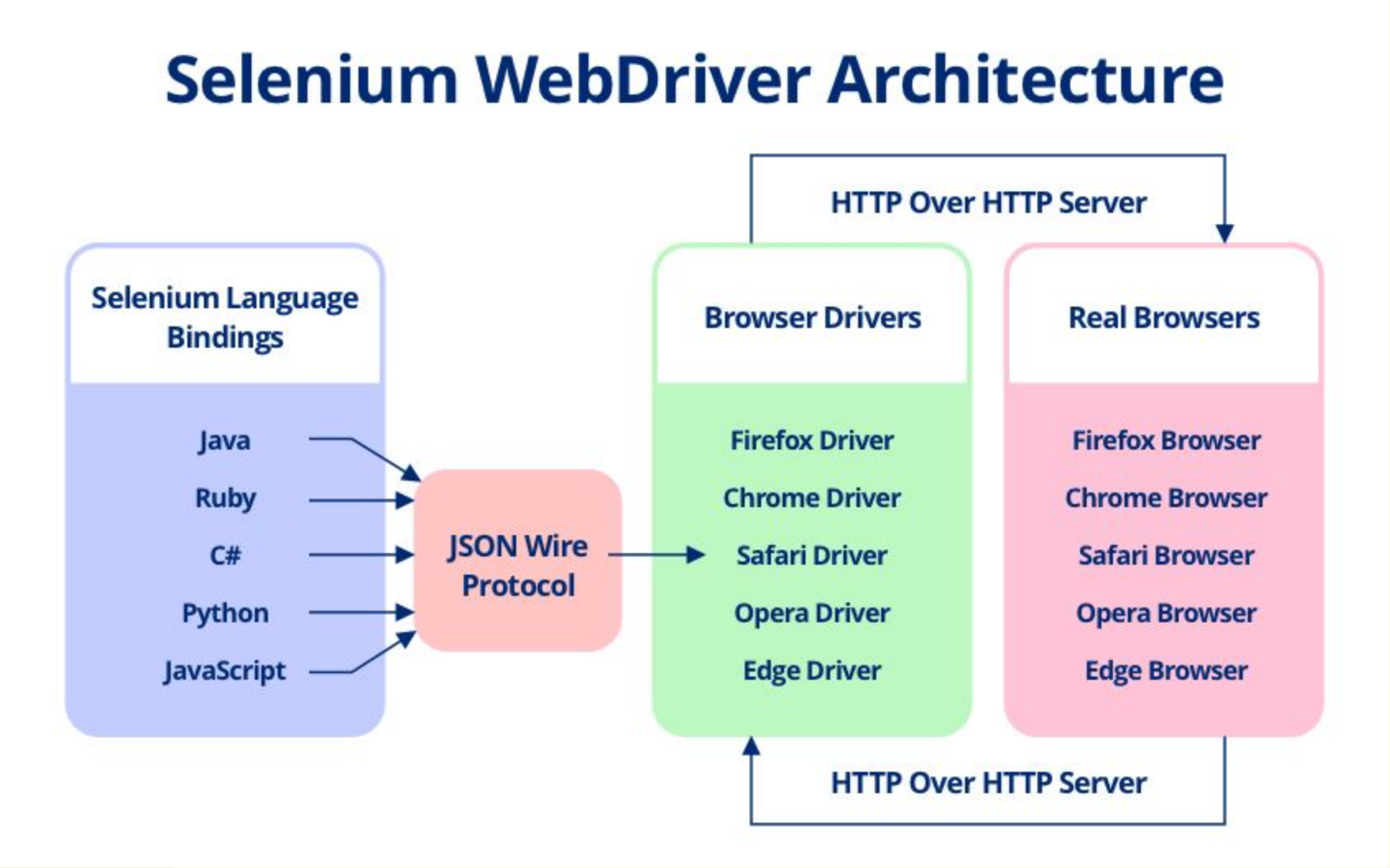
What is Selenium

"Selenium automates web browsers"

https://www.seleniumhq.org/

- An open-source automated testing suite for web applications
- Selenium supports **all major browsers** (Chrome, Firefox, Internet Explorer), works on **many operating systems** (Linux, OS X, Windows) and can be used with **many programming languages** (Python, C#, Java, Haskell)
- 3 flavors: WebDriver, IDE, Grid

How it works


Selenium WebDriver Architecture
HTTP Over HTTP Server
Selenium Language Bindings
Java
Ruby
C#
Python
JavaScript
JSON Wire Protocol
Browser Drivers
Firefox Driver
Chrome Driver
Safari Driver
Opera Driver
Edge Driver
Real Browsers
Firefox Browser
Chrome Browser
Safari Browser
Opera Browser
Edge Browser
HTTP Over HTTP Server

Selenium vs Scrapy

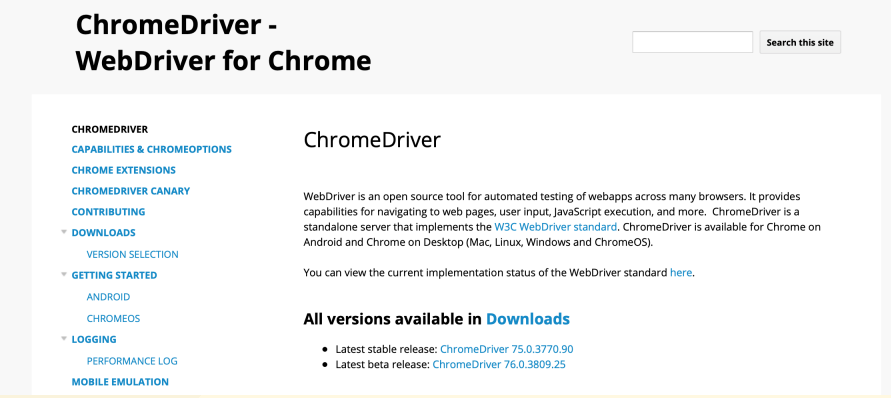
- Selenium built for browser automation, not for scraping, therefore slower performace to be expected
- Less steep learning curve
- Easier to scrape dynamic pages with JavaScript elements

Download and installation

1. Add Selenium to your project:

```
pip install selenium
```

2. Download your WebDriver and ensure it is in your $PATH
   http://chromedriver.chromium.org



First steps

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

# path to ChromeDriver
chromedriver = "/usr/local/bin/chromedriver"

# create an instance of Chrome Webdriver
driver = webdriver.Chrome()

# navigate to a url
driver.get("http://www.google.com/search?q=pymug");

# wait 5 seconds and close the browser
time.sleep(5)
driver.close()
```

Finding elements on the page

- Selenium provides several locators to interact with UI elements:
  - ID
  - Name
  - Link Text
  - CSS Selector
  - DOM (Document Object Model)
  - XPath

Using locators

```
# find an element using XPath
elem = driver.find_element_by_xpath(
    "//div[@class='r'][1]/a"
)

# interacting with an element
elem.click()
elem.send_keys("pymug")
elem.send_keys(Keys.RETURN)
```

Scraping dynamic webpages

Our target:

Scraping dynamic webpages

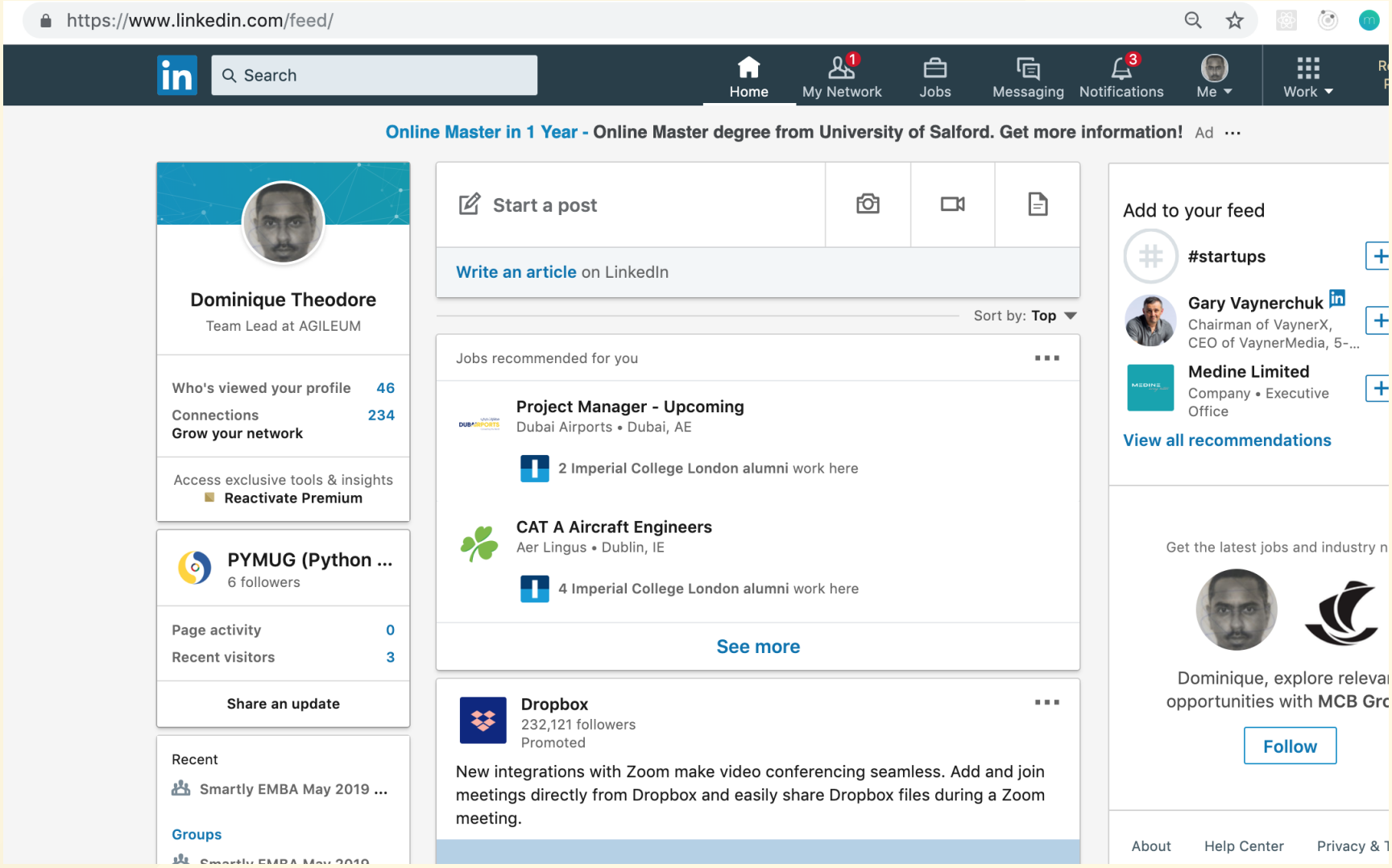
```
                <td width="225">ICT/BPO</td>
                <td width="175">GLOBAL EDGE SOFTWARE LTD</td>
                <td width="125">Mauritius</td>
                <td width="100">28/06/2019</td>
                <td align="center"><img src="https://www.mauritiusjobs.mu/images/icon_view.png" id="view_job17063" alt="17063" title="Show Details" w
height="20"></td>
            </tr>

            <tr><td colspan="7" class="hidden-td">
                <div class="show_details" id="17063">
                <b>Web Developer</b>
                    <table class="job_details" width="100%">
                    <tr>
                        <td width="300">Employer</td>
                        <td>GLOBAL EDGE SOFTWARE LTD</td>
                    </tr>
                    <tr>
```

```
<div class="main-content">
    <h1>Jobs Search</h1>

    <script type="text/javascript">function jobdetails(val){$("#"+val).slideToggle();}
function applyjob(val)
{$(this).scrollTop(0);$('#popup').show();$('#popup_content').show();$.ajax({type:"POST",url:"ht
on(msg){document.getElementById("popup_content").innerHTML=msg;}});}
```

Scraping dynamic webpages

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

chromedriver = "/usr/local/bin/chromedriver"

driver = webdriver.Chrome()
driver.get("https://www.mauritiusjobs.mu/jobsearch");

input = driver.find_element_by_name("search_by_keyword")
input.send_keys("developer")
input.send_keys(Keys.RETURN)

result = driver.find_element_by_xpath(
    "//img[contains(@id, 'view_job')]")
result.click
```

Infinite scrolling pages



Infinite scrolling pages

- Use expected conditions to wait for the page to load and prevent Element Not Visible Exception

Selenium Waits

```
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
```

- Selenium can execute Javascript and scroll the page for us

```
driver.execute_script("window.scrollTo(0,
    document.body.scrollHeight);")
```

Thank you!