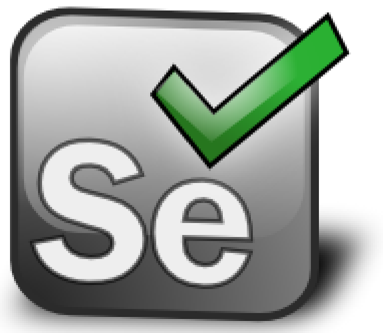
Pro Tips for Selenium Configuration

Let your crawler works correctly and efficiently.

petertc May 20, 2019 · 3 min read



From my previous post, you already know how to setup Selenium and ChromeWebDriver for web scraping. Now let's jump into the rabbit hole to see how to build a robust crawler.

The Basic Crawler

In the beginning, your crawler with ChromeWebDriver may look like this:

```
from selenium import webdriver


driver = webdriver.Chrome()
driver.get('http://www.google.com')
...
```

It should work in your box, but you may encounter issues when getting serious. You may deploy the crawler either on a server or a container. The environment is a problem. You also want your crawler can work correctly and efficiently.

webdriver.Chrome() Cheat Codes

Here I collect four cheat codes to get rid of these common issues. The first two code snippets address environmental issues, and the others are about correctness and efficiency.

1. Run a Crawler on a Server

The server here means that there is no GUI, no X window available in the box. Modern browsers usually have to be run with a GUI environment. Many articles, such as that one, guide you to use XVFB to simulate such an environment. The great news is Headless Chrome is shipping in Chrome 59. Now you can run a crawler on a server with this code snippet:

```
opts = webdriver.ChromeOptions()
opts.add_argument('--headless')
driver = webdriver.Chrome(options=opts)
```

2. Run a Crawler in the Container or Serverless Way

It's time to cloud-native! Many people will like to run their fancy apps with docker or k8s instead of an annoying Linux server (Not for me.) When you run a crawler in docker, you may see some weird errors pop-up, such as:

```
Running as root without --no-sandbox is not supported.
```

or

```
Creating shared memory in /dev/shm/.org.chromium.Chromium.JwBSnH
failed: No such file or directory (2)
```

Try to use this cheat code to address the issue:

```
opts = webdriver.ChromeOptions()
opts.add_argument('--no-sandbox')
opts.add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome(options=opts)
```

3. Change User-Agent

Website administrators can distinguish your crawler from normal users by the User-Agent. For example, this is my default User-Agent:

```
In [6]: driver.execute_script("return navigator.userAgent")
Out[6]: 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Ubuntu Chromium/73.0.3683.86 HeadlessChrome/73.0.3683.86
Safari/537.36'
```

Normal users never have HeadlessChrome. That’s the point.

Website administrators may block your crawler according to information in User-Agent. To get rid of that, you can fake it by the following options:

```
opts = webdriver.ChromeOptions()
opts.add_argument('--user-agent=""Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/74.0.3729.157 Safari/537.36""')
driver = webdriver.Chrome(options=opts)
```

The above code makes your crawler as a typical Windows 10 chrome user. It's harder to be distinguished now.

4. Do crawling more efficiently

Web crawling is a resource-intensive job. Dinuduke shares some tips which can minimize time consumption:

```
opts = webdriver.ChromeOptions()
prefs = {"profile.managed_default_content_settings.images":2,
         "profile.default_content_setting_values.notifications":2,
         "profile.managed_default_content_settings.stylesheets":2,
         "profile.managed_default_content_settings.cookies":2,
         "profile.managed_default_content_settings.javascript":1,
         "profile.managed_default_content_settings.plugins":1,
         "profile.managed_default_content_settings.popups":2,
         "profile.managed_default_content_settings.geolocation":2,
         "profile.managed_default_content_settings.media_stream":2,
}
```

```
opts.add_experimental_option("prefs",prefs)
driver = webdriver.Chrome(options=opts)
```

Briefly, the code above asks the browser loads useful information for web crawling and skip those are only valuable for human beings.

Final Word

Even though numinous posts and articles are talking about Selenium and web crawling on the Internet, I still spent much time to google, troubleshooting, for just getting a start. I hope this article can save you time and enjoy crawling painlessly!

Selenium Web Crawling Web Crawler Web Scraping Python