Web Scraping Framework Review: Scrapy VS Selenium

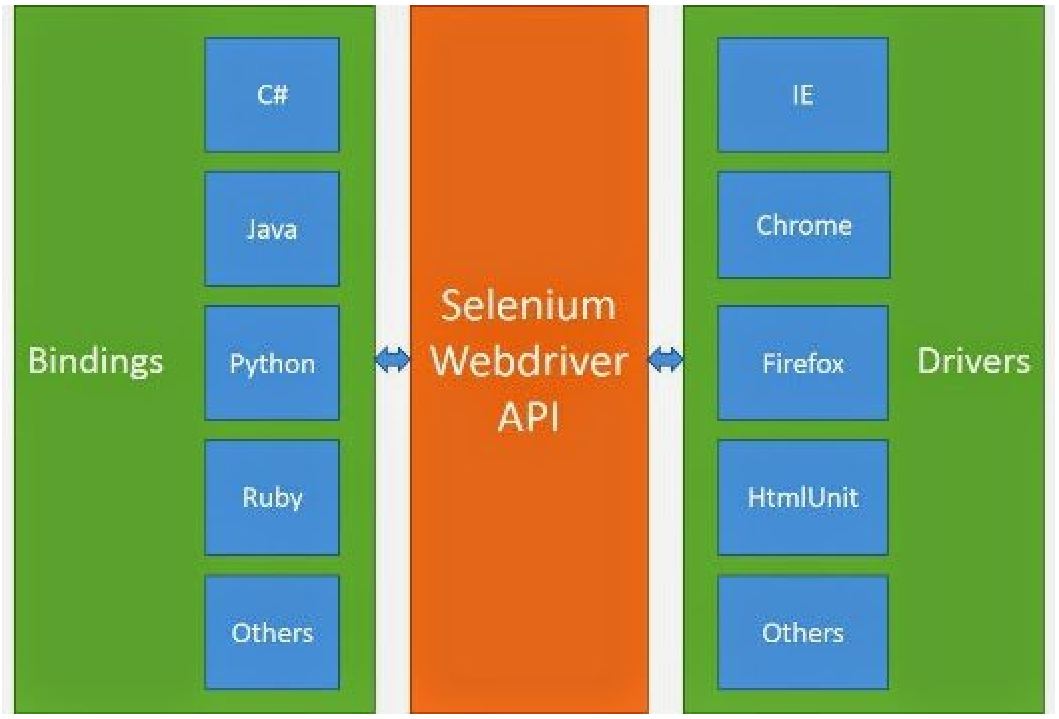
Last updated on Feb 25 2019 by Michael Yin

Introduction:

This is the #11 post of my Scrapy Tutorial Series, in this Scrapy tutorial, I will talk about the features of Scrapy and Selenium, Compare them, and help you decide which one is better for your projects.

Talk About Selenium

Selenium is a framework which is designed to automate test for web applications. It provides a way for developer to write tests in a number of popular programming languages such as C#, Java, Python, Ruby, etc. The tests writen by developer can again most web browsers such as Chrome, IE and Firefox.



As you can see, you can write Python script to control the web brwoser to do some work automatically. For example, you can make browser visit craigslist, click target elemnt or navigate to the target page, get the html source code of page.

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

driver = webdriver.Firefox()
driver.get("http://www.python.org")
assert "Python" in driver.title
elem = driver.find_element_by_name("q")
elem.send_keys("selenium")
elem.send_keys(Keys.RETURN)
assert "Google" in driver.title
driver.close()
```

From the code above, you can see, the API is very beginner-friendly, you can easily write code with Selenium. That is why it is so popular in developer community. Even Selenium is mainly use to automate tests for web applications, it can also be used to develope web spider, many people has done this before.

Talk About Scrapy

Scrapy is a web crawling framework for developer to write code to create `spider`, which define how a certain site (or a group of sites) will be scraped. The biggest feature is that it is built on Twisted, an asynchronous networking library, so Scrapy is implemented using a non-blocking (aka asynchronous) code for concurrency, which makes the spider performance is very great.

For those who have no idea what is `asynchronous`, here is a simple explanation.

> When you do something synchronously, you wait for it to finish before moving on to another task. When you do something asynchronously, you can move on to another task before it finishes.

Scrapy has built-in support for extracting data from HTML sources using XPath expression and CSS expression.

Which One Should You Choose?

The two Python web scraping frameworks are created to do different jobs. `Selenium` is only used to automate web browser interaction, `Scrapy` is used to download HTML, process data and save it.

When you compare `Selenium` vs `Scrapy` to figure out what is the best for your project, you should consider following issues.

Javascript

You should use some tool such as Dev Tool from Chrome to help you figure out how the data is displayed on the dynamic page of target site. If the data is included in html source code, both frameworks can work fine and you can choose one as you like. But in some cases the data show up after many `ajax/pjax` requests, the workflow make it hard to use Scrapy to extract the data. If you are faced with this situation, I recommend you to use Selenium instead.

Data Size

Before coding, you need to estimiate the data size of the extracted data, and the urls need to visit. Scrapy only visit the url you told him, but Selenium will control the browser to visit all js file, css file and img file to render the page, that is why Selenium is much slower than Scrapy when crawling.

If the data size is big, Scrapy is the better option because it can save you a lot of time and time is a valuable thing.

Extensibility

The architecture of `Scrapy` is well designed, you can easily develop custom middleware or pipeline to add custom functionality. Your `Scrapy` project can be both robust and flexible. After you develop several Scrapy projects, you will benefit from the architecture and like its design because it is easy to migrate from existing Scrapy spider project to another one. You can check this artcile to see how to quickly save the scraped data into Database by using Scrapy pipeline without modifying the code of spider. Scrapy Tutorial #9: How To Use Scrapy Item

So if your project is small, the logic is not very complex and you want job done quickly, you can use `Selenium` to keep your project simple. If your project needs more customization such as proxy, data pipeline, then the `Scrapy` might be your choice here.

Ecosystem

Very few people have talked about this before when comparing web scraping tools. Think about why people like to use Wordpress to build CMS instead of other frameworks, the key is `ecosystem`. So many themes, plugins can help people quickly build a CMS which meet the requirement.

Scrapy have so many related projects, plugins on open source websites such as Github, and many discussions on StackOverflow can help you fix the potential issue. For example, if you want to use proxy with your spider project, you can check a project `scrapy-proxies` help you send HTTP requests using random proxy from list. All you need is just changing some settings.

Best Practise

If you like Scrapy and you also want it to understand JavaScript, there are also some options for you.

1. You can create new instance of webdriver from Selenium in `parse` method of Scrapy spider, do some work, extract the data, and then close it after all work done. You should remember to close it or it might cause some problem such as memory.

2. Scrapy has offical project(I really like its ecosystem) called scrapy-splash to provides Scrapy and Javascript integration.

If you are Selenium's fan, and want spider to run quietly, you can try to use Phantomjs, a headless browser. I like to develop spider using Selenium with ChromeDriver because it is easy to debug, when I am done, the spider would run with phantomjs in terminal.

Conclusion

So which one is better web scraping framwork? There is no solid answer, the answer depends heavily on the actual situation. Below is a quick reference table.

Framework	Selenium	Scrapy
Javascript Support	Support javascript very well	It is time consuming to inspect and develop spider to simulate ajax/pjax requests
Data Size	Good option for small data set	Work fine on big data set
Extensibility	Not very easy to extend the project	You can easily develop custom middleware or pipeline to add custom function, easy to maintain.
Ecosystem	Not many related projects or plugins	Many related projects, plugins on open source websites such as Github, and many discussions on StackOverflow can help you fix the potential issue.

In short, **If the job is a very simple project, then `Selenium` can be your choice. If you want a more powerful and flexible web crawler, or you indeed have some experience in programming, then `Scrapy` is definitely the winner here.** What is more, if you want your Scrapy spider to understand the javascript, just try methods mentioned above.

If you are also interested in BeautifulSoup, a great web scraping framework in Python world, you can take a look at Scrapy VS Beautiful Soup