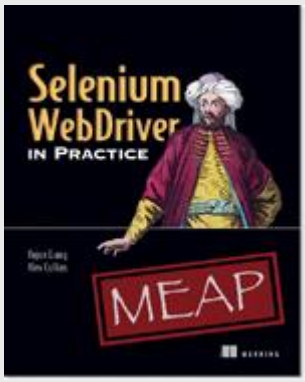

What is WebDriver?

By Yujun Liang and Alex Collins

In this article, excerpted from Selenium WebDriver in Practice, we will introduce WebDriver, what it is, how it works, and reasons for choosing it.

Selenium WebDriver automates web browsers. It sits in the place of the person using a web browser. Like a user, it can open a website, click links, fill in forms, and navigate around. It can also examine the page, looking at elements on it and making choices based on what is sees.

The most common use case for WebDriver is automated testing. Until recently, to run a regression test on your website, you'd need to have a set of scripts that would have to be manually executed by developers or QAs. Any reports would need to be manually collated too. This can be both time-consuming and costly. Instead, WebDriver can be used to execute those scripts, and automatically gather reports on how successful they were, at the push of a button. Each subsequent execution will be no more expensive than the first.

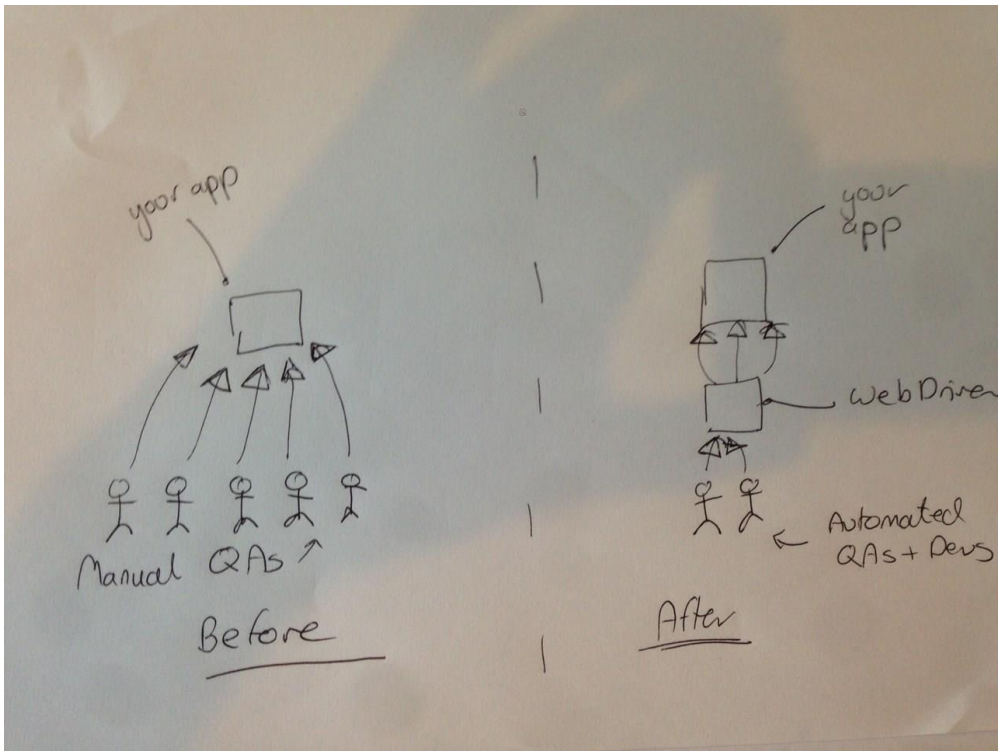

Figure 1 Before WebDriver

Long gone are the days when you needed to create one version on your website for the pervasive and notoriously standards non-compliant Internet Explorer 6, and another for other browsers. While most modern browsers are much more consistent in their behavior, the way a web page looks or acts can still greatly vary as the number of different browsers, operating system, and platforms in common use has increased. You can still have a high-value customer complain that they can't access your site. Historically, the only way to mitigate this was to have army of QAs manually test on a variety of different configurations, a time-consuming and costly process. WebDriver can run tests on different operating systems and different browser configurations, and in a fraction of the time of a human being. Not only that, you can use it to run them much more consistently and reliably than a manual tester.

Applications and websites provide useful services, but sometimes these are only accessible by web pages. Another use case for WebDriver is to make those pages accessible to applications via WebDriver. You might have an administration application written several years ago and a client or PO has asked for some actions on it to be automated. But maybe no one

knows where the source code is. It might be much easier to use WebDriver to automate this task.

How WebDriver works

WebDriver works in all major browsers and with all major programming languages. How is this possible? Well, WebDriver has several interacting components:

1. A web browser.
2. A plugin or extension to the browser that lives inside the browser, which itself contains a server that implements the WebDriver JSON API.
3. A language binding (in our case Java) that makes HTTP requests to that API.

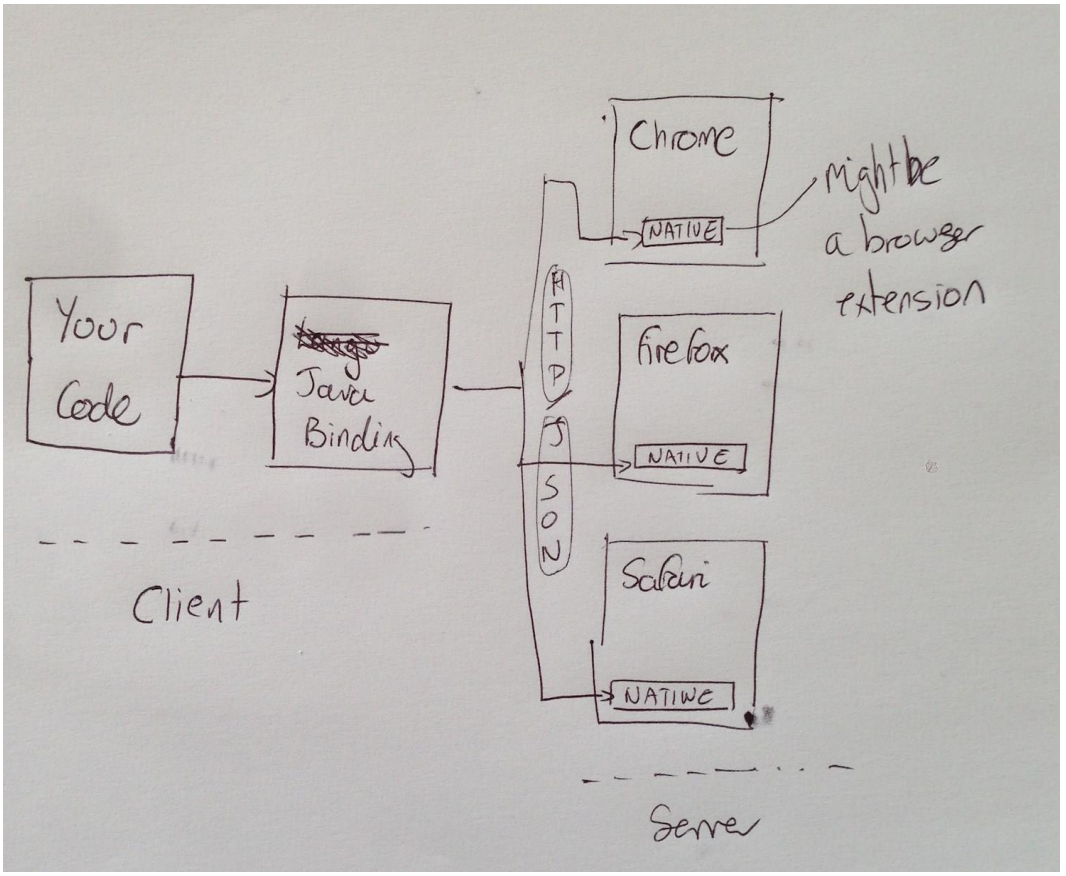

Figure 2 Web driver diagram

When you start code that uses WebDriver, it will open up the browser, which in turn starts the plugin. You can then send requests to perform the actions you want, such as clicking on links or typing text. As a plugin only needs to implement the JSON API, people have written plugins for all major browsers. To use a browser that has a plugin, you just need to implement a client to the JSON protocol.

This means that all the major browsers and all the major programming languages support WebDriver.

The plugin can usually be seen in the browser's preferences, such as in figure 3.

Figure 3 Safari Extensions panel

Why choose WebDriver?

There a number of great reasons to choose WebDriver:

- WebDriver can run browsers *locally and remotely* with minimal configuration changes.
- WebDriver is *supported by major browser vendors*: both Firefox and Chrome are active participants in WebDriver's development.
- WebDriver more closely mimics a user. Where possible it uses *native events* to operate, to make it accurate and stable.
- WebDriver is *Open Source Software (OSS)*. This means that it is both free and is supported by an excellent community.
- WebDriver supports all major operating systems such as *OS X*, *Windows*, and *Linux*. It also has support for *Android* and *iOS*.
- WebDriver is going to become a *W3C* standard, so you can expect that it will be supported for a long time.
- WebDriver doesn't suffer from some of the problems that Selenium 1.0 suffered from, such as with uploading files, or handling pop-ups.

- WebDriver has a more *concise syntax* than Selenium 1.0, making it faster to write code.

What WebDriver cannot do

WebDriver provides a way to control a web browser, but that's all. When you buy a new car, you get a manual that will tell you how to operate the radio and how change the oil. But that manual won't tell you the best place to get your car serviced, or teach you how to drive. Like driving a car, there are things you must do for yourself. Here are some things WebDriver does not do:

- WebDriver doesn't have the control of the timing of the elements appearing on web page. Some might appear later and you'll need to handle this yourself.
- WebDriver does not know when things have changed on the page, so you can't ask it to tell you when things have changed.
- WebDriver doesn't provide many utilities for writing your code. You need to write these yourself.
- WebDriver doesn't provide built-in support for page elements that are composed of other elements, such as JavaScript calendars.
- WebDriver does not provide a framework to write your code in. JUnit is a natural choice.
- WebDriver doesn't manage the browser. For example, you need to clean up after you have used it.
- WebDriver won't install or maintain your browsers. You need to do this yourself.

So that, in a nutshell, is what WebDriver is, how it works, and reasons for choosing it.