WonderProxy

presents

Automated Testing of Your Corporate Website from Multiple Countries with Selenium

Contents

Summary

Because of the complexities involved in testing large corporate websites and ecommerce stores from multiple countries, test automation is a must for every web and ecommerce team. Selenium is the most popular, straightforward, and reliable test automation framework with the largest developer community on the market. This white paper details how Selenium can be integrated with a worldwide proxy network to verify website availability, performance, and correctness on a continuous basis.

Introduction

Modern enterprise web development teams face a number of challenges when they must support access to their website from multiple countries. These challenges include verifying availability, verifying performance, and verifying content correctness on a daily basis. Website content is presented in different languages, website visitors use different browsers and operating systems, and ecommerce carts must comprehend different products and currencies.

Because of these complexities involved, instituting automated tests via a test automation framework is the *only* feasible method of verifying all of these aspects in a repeatable and regular fashion.

Why automate tests?

Every company tests its products before releasing them to their customers. This process usually involves hiring quality assurance engineers and assigning them to test the product manually before any release. Manual testing is a long process that requires time, attention, and resources in order to validate the products' quality.

The more complex the product is, the more important, complex, and time-consuming the quality assurance process is, and therefore the higher the demand for significant resources.

Experience shows that many of the tests quality assurance engineers perform are repetitive, and therefore could be performed automatically and quickly, with minimal supervision. Automated tests give the quality assurance engineers time to focus on other tests that they simply didn't have enough time to perform in the past.

Just think: With automated tests, you could increase the coverage for your product's quality testing, and spend less time and resources in the process!

Why Selenium?

These days Selenium is the most popular and most reliable test automation framework out there. It gives us the ability to automate browsers to perform everything a real user can do!

You can use Selenium with most modern programming languages and benefit from the huge community of Selenium developers online, who provide loads of information and support.

This white paper details how a team can implement test automation of a corporate website or ecommerce store from multiple countries, using Selenium in combination with a worldwide proxy network.

The Challenges

SLA categories	Technical and organizational challenges
Availability	Configuration management with multiple teams in multiple time zones
Performance	Hosting, latency, CDN differences
Correctness	Multi-product, multi-language, multi-currency

Figure 1 Enterprise Website Multi-Country Testing Challenges

Availability

As website owners, developers or managers we would like to be able to monitor how our website performs in different countries. Sometimes, our website or ecommerce store might not be fully available in different countries due to regional DNS outages, network routing errors, and low internet quality. Using automation in combination with a worldwide proxy network helps us monitor our website's availability from different countries. Availability monitoring can help us make operational decisions, such as blocking parts of the website in countries that we discover have poor internet connectivity, or developing a lighter version of the website suitable for weaker internet connections. You might also choose to display relevant error messages, or offer alternative content on the spot.

We must remember that many countries block content on the internet (such as images, videos, certain text, etc.). Global monitoring helps make sure that your application works flawlessly everywhere, and if any part of it gets blocked by the local internet providers, you are aware of it as soon as possible.

Performance

One of the most important aspects of every website is its performance. Even if your website has the best features on the market, if it just performs too slowly, your users will be unsatisfied and eventually avoid using it.

Research shows that your website load time has a great impact on your users' satisfaction.[1] According to this research, a delay of only 2 seconds resulted in a 4% decrease in customer satisfaction and a 4.3% loss in income per customer.

We want to make sure that any user of our website at any part of the world, no matter how far it is from your servers, will be able to use your product with the best performance with the lowest latency. In addition, we want to ensure that if there is a performance hit in a certain country or region in the world, we know about it as soon as possible, and can handle it appropriately.

Correctness

If you are delivering localized content to your users based on their location, it is mandatory to verify that they are able to view your website in their language, see product prices in their local currency, and see the correct content for their location. For many companies, it is also very important to verify that their users see localized ads. An advertisement for a French local business for a user in the United States is not likely to be clicked. Similarly, location-specific offers (based on a local state sports team for example) can also be a major source of income.

Many websites are designed to be used by people from all around the world, each one in a different time zone. In case there is a collaboration between the website users while each one of them is located in a different country it is mandatory to make sure that the website will perform flawlessly with any combination of time zones used between the collaborating users. Certain bugs could be caused due to problems in time zone difference calculations and we can discover them only through extensive testing from all possible user locations.

Another very important issue is content blocking. You may need to block certain content (like images, videos, music) in some counties for legal or regulatory reasons, in order to avoid disputes with the local authorities or with your content providers.

Meeting the Challenges with Automation

These are the main challenges we face when trying to build a high-quality international website or ecommerce store. Unfortunately, while most companies

---

[1] *What is a CDN?* (n.d.). Cloudflare. Retrieved March 9, 2020, from https://www.cloudflare.com/learning/cdn/what-is-a-cdn/

understand the need to test their websites from different countries in order to overcome the challenges we discussed, they find it almost impossible to test more than a few locations due to the enormous resources it would require.

How much work are we talking about, exactly?

You are probably going to need to manage a few teams in different parts of the world. Each team is probably going to have at least a few Quality Assurance Engineers in it, all these teams will have to collaborate with each other (to maintain a single source of truth for test cases and bug reports, and to make sure their testing strategies work well).

On top of the human resources, there are the costs of maintaining a worldwide proxy network yourself. You'll need to manage servers (either with a cloud provider, paying for every hour each one of your servers is up and running, or having on-prem service, which is much more expensive and hard to manage), configure the network, and develop a networking monitoring system. All these are probably going to require you to hire at least one experienced DevOps Engineer.

This amount of work can be too much to invest in manual localization testing, and the resulting lack of test coverage eventually leads to increased costs for website maintenance. Customers discover geolocation-related bugs themselves, and you may lose their income.

This is where automation comes to help - automating your tests with Selenium and a reliable worldwide proxy network (such as WonderProxy) will cover all the aspects of geolocation testing in a significantly shorter time with minimal resources required. And most importantly - while your automatic tests are already written you can just run them on every release of your application.

Components of a Solution

It is clear that in order to automate the testing of our website or ecommerce store from different countries we need a good, easy, and reliable solution.

In this section, we will discuss the components of the only practical solution to this problem: writing automated tests in a programming language of your choice (Python, Java, C#, PHP, Ruby, JavaScript, and Kotlin), and running them with a combination of the Selenium automation framework, a web browser called PhantomJS, and a worldwide proxy network.

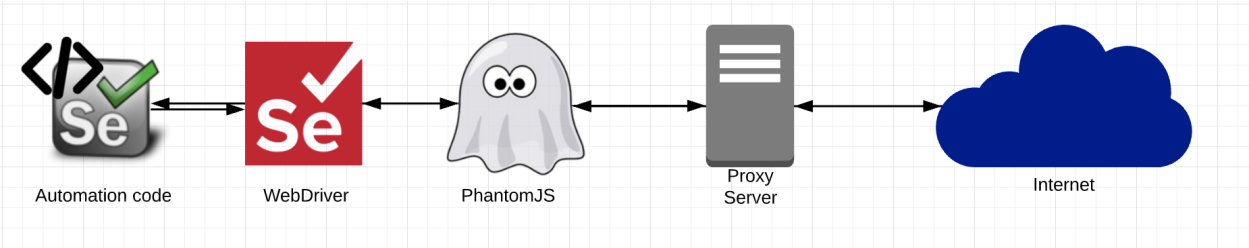

Figure 2 How the components fit together

Now let's break this solution into smaller bits and explain each one:

Selenium

Selenium is the most popular browser automation library.

Primarily, it is used for automating web applications for testing purposes, but is not limited to that use case. The tests can be run over any modern web browser including Chrome, Firefox, Edge, Safari and Internet Explorer.

You can use the Selenium framework with most programming languages, including Python, Java, C#, PHP, Ruby, JavaScript, and Kotlin.

PhantomJS

PhantomJS is a headless browser scriptable with JavaScript. A headless browser is similar to any other browser like Chrome or Firefox, except it does not have a graphical user interface (GUI). Instead, it runs in the background and performs much faster.

PhantomJS helps developers automate the process of testing their websites without a need for any sort of browser GUI. It offers native support for various web standards: DOM handling, CSS selectors, JSON, Canvas, and SVG.

PhantomJS can run on all major operating systems: Windows, macOS, Linux, and other Unices.

WebDriver

Selenium's WebDriver is a remote control interface that enables control of browsers. The WebDriver accepts commands from your test code and sends them to the browsers it controls:

> *This is implemented through a browser-specific driver [Chrome Driver, Firefox Driver, PhantomJS], which sends native commands to a browser and retrieves results. Most browser drivers launch and access a browser application [except PhantomJS that runs in the background].*[2]

A Proxy Network

A standard proxy network consists of 3 different parts:

1. Your computer
2. The proxy server
3. Your target website

The proxy server serves as a mediator between your computer and the website you want to test. The server uses different computers in various locations in order to obtain web pages or files without exposing your real location to the websites' host.

In the following example we'll demonstrate what happens when you try to access www.google.com through a proxy network (without getting too technical):

1. You type https://www.google.com in your web browser.

---

[2] *Selenium (software).* (2020). Wikipedia. Retrieved March 9, 2020, from https://en.wikipedia.org/wiki/Selenium_(software)

2. The request is forwarded to the proxy server.
3. The proxy server receives the request.
4. The proxy server passes the request to http://www.google.com.
5. At this point, Google sees the proxy server as the visitor instead of seeing your computer as the visitor. If your computer is in Canada but the proxy server is located in Belgium, Google will see you like a Belgian visitor.
6. The proxy server retrieves the response from Google.
7. The proxy server passes the response to your computer. This is where you'll see a localized version of Google for Belgian users.

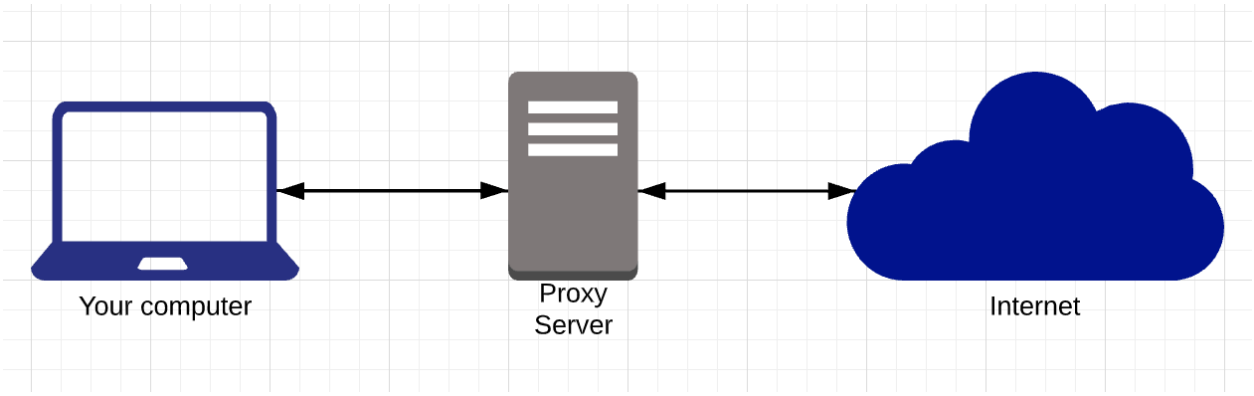

Figure 3

Steps

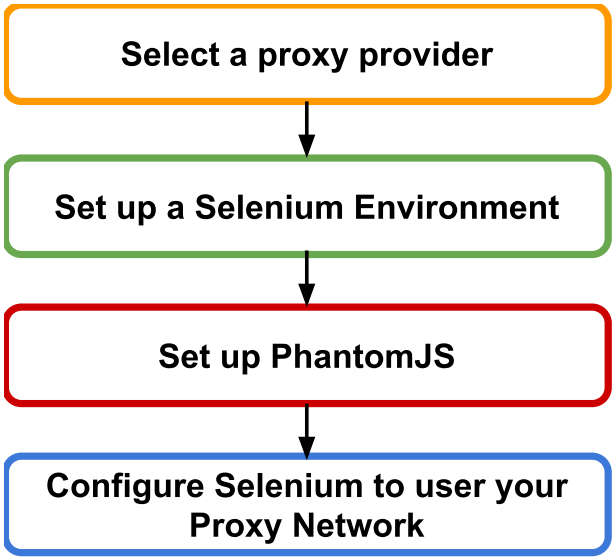

Figure 4

In this section, we are going to cover all the essential steps required to set up an environment for testing your website from different countries.

There are four fundamental steps in this process: Selecting the right proxy provider, setting up a Selenium environment, setting up PhantomJS and finally, configuring Selenium to use your proxy network.

Let's see how it's done:

A. Select a Proxy Provider

Selecting the right proxy provider suitable for your needs will be your first and most important step.

Recent research on over 6.2 million proxy service providers' IP addresses, focused on the security aspects of some of the leading proxy providers, states that these proxy providers pose new threats to internet users. [3]

[3] Mi, X., Feng, X., Liao, X., Liu, B., Wang, X., Qian, F., Li, Z., Alrwais, S., Sun, L., & Liu, Y. (2019). Resident Evil: Understanding Residential IP Proxy as a Dark Service. *2019 IEEE Symposium on Security and Privacy (SP)*. https://doi.org/10.1109/sp.2019.00011

These proxy providers offer residential IP addresses...

> *... as intermediaries to circumvent the restrictions imposed by target services, for purposes such as aggressive resource access (e.g., registering multiple accounts), data scraping, and others.*

All these providers control a large number of *shared hosts*. These residential proxies...

> *... can outperform conventional public proxies or even anonymity networks to help their clients masquerade as clean and benign sources to communicate with the targets. Such communication may violate the target's service terms at the very least ... and is likely associated with more sinister events such as the aforementioned DDoS, due to the permissiveness of the ... providers in terms of what can be done through their proxies.*

According to the study, there is no proof that all the hosts are willingly participating in the service and it is impossible to determine whether these proxies are malicious or not; however, "2.20% percent of all IPs researched are reported by public blacklists or emerging threat intelligence platforms."

Furthermore, it was discovered that the major proxy providers resell their services to other providers, which means that their resources are shared and might be abused by the other providers.

In total, 237,029 of the residential IPs studied were identified as "internet of things" (IoT) devices, including "web camera, DVR, and printer."

Another discovery was that 2.32% of one of the researched providers' residential IPs "were hosting malicious content or having suspicious domains resolved to them while acting as proxies."

This study helps us understand that many proxy network providers are simply not secure enough to use at the corporate level. You can't know who is providing the proxy server, so you are exposing your company to possible security risks.

We can't stress enough how important it is to make sure that your proxy provider is reliable and secure, so you can perform your tests and protect your data. Even one security breach is enough to shut down a whole company.

B. Set up your local Selenium environment

Initializing a local Selenium server is nice and easy, so we'll do that first.

1. Go get the stand-alone server (packaged as a Java archive) from Selenium's download page:

```
$ wget https://selenium-release.storage.googleapis.com/3.7/selenium-server-standalone-3.7.1.jar
```

2. Spin it up with your local Java installation:

```
$ java -jar selenium-server-standalone-3.7.1.jar
```

It starts with some sensible defaults, but you can configure them to your heart's content! Run the server with a `-h` flag for details.

C. Get PhantomJS

PhantomJS is a "headless" browser engine that acts just like a regular browser without the graphical user interface. It's ideal for testing website behavior and content when you're not overly concerned with styling and display.

1. Follow the official download instructions. (On bigger Linux distributions, you can generally use the native package manager.)

2. Make sure the `phantomjs` executable is in your `$PATH`:

```
$ sudo ln -s /path/to/phantomjs /usr/local/bin/phantomjs
```

Wait a minute, why are you using all this old tooling?

Great question! You're right: the PhantomJS project has been suspended, and Selenium 3.7.1 released in November 2017. Why aren't we using the latest and greatest?

Sadly, modern tooling doesn't support authenticated proxies, proxies that require a username and password. Neither Headless Chrome nor Headless Firefox allow Selenium to configure proxy authentication. PhantomJS does, but it's only supported by older versions of Selenium. That's what we're using here.

D. Configure Selenium to use a proxy network

Writing Selenium tests is extensively covered in books, webinars, articles, and videos, so we won't go in-depth on the topic here. Instead, we'll focus on integrating a proxy network using Selenium's `DesiredCapabilities` interface. `DesiredCapabilities` allows testers to customize browser configurations. In our case, we will customize PhantomJS to proxy its connections through a WonderProxy server, but these instructions will work for any authenticated proxy server.

The PHP and Python demo tests read your WonderProxy credentials from the local environment, so set that up first:

```
$ export WONDERPROXY_USER=yourusername
$ export WONDERPROXY_PASS=yourpassword
```

In the code samples below, we pass a proxy server (like `denver.wonderproxy.com:11000`) into the `proxied()` method as an argument. Then, `proxied()` plugs our proxy and proxy credentials into Selenium's PhantomJS driver, using `DesiredCapabilities`. Finally, `proxied()` returns the customized driver, which will direct all its requests through the proxy server.

Code sample in PHP

```
    protected function proxied($proxy) {
        $capabilities = DesiredCapabilities::phantomjs();
        $capabilities->setCapability(
            'phantomjs.cli.args',
            [
                "--proxy=$proxy",
                '--proxy-type=http',
                '--proxy-
auth='.getenv('WONDERPROXY_USER').':'.getenv('WONDERPROXY_PASS')
            ]);
        return RemoteWebDriver::create($this->selenium, $capabilities);
    }
```

Code sample in Python

```
    def proxied(self, proxy):
        capabilities = DesiredCapabilities.PHANTOMJS.copy()
        capabilities['phantomjs.cli.args'] = [
            '--proxy=' + proxy,
            '--proxy-type=http',
```

```
            '--proxy-auth=' + evar.get('WONDERPROXY_USER') + ':' +
evar.get('WONDERPROXY_PASS')
        ]

        return webdriver.Remote(
            command_executor=self.selenium,
            desired_capabilities=capabilities
        )
```

Get testing!

Now that we've configured Selenium's WebDriver to shunt connections through a proxy server, we can write some tests using the proxied driver.

Code sample in PHP

The PHPUnit testing framework supports "parameterized" tests (tests that can be run multiple times with varied input) out of the box with its `@dataProvider` notation. In this sample, the `userLocations()` method will be the input source for our test. It includes three sets of input, each containing the proxy to be tested and the expected output.

```
    public function userLocations() {
        return [
            ['albuquerque.wonderproxy.com:11000', 'Albuquerque'],
            ['toronto.wonderproxy.com:11000', 'Toronto'],
            ['vancouver.wonderproxy.com:11000', 'Vancouver'],
        ];
    }
```

The test itself grabs a proxied driver, then loads up a web page and checks the `user-city` element.

```
    /**
     * @dataProvider userLocations
     */
    public function testUserLocation($proxy, $expected) {
        $this->driver = $this->proxied($proxy);

        $this->driver->get($this->url);
        $search = $this->driver->findElement(WebDriverBy::id('user-city'));
        $this->assertContains($expected, $search->getText());
    }
```

The element should read `Toronto` for the `toronto` proxy, `Vancouver` for the `vancouver` proxy, etc.

Code sample in Python

Python's unittest framework does not include parameterized test support, but the gist is the same as above: The test creates a proxied driver, then checks the `user-city` element against the proxy.

```
    def testUserLocationAlbuquerque(self):
        self.driver = self.proxied('albuquerque.wonderproxy.com:11000')
        self.driver.get(self.url)
        search = self.driver.find_element_by_id('user-city')
        self.assertIn('Albuquerque', search.text)
```

In this section, we have covered the four fundamental and simple steps required to get your ready to start testing your application from anywhere in the world using a global proxy network and Selenium.

As you just saw, this is a straightforward and practical approach that will get you ready to get running in less than 20 minutes!

Working Demo

There's no better way to see how all the pieces connect than a working demo.

We've prepared a working demo of these pieces that you can download from GitHub and modify to meet your requirements.

Conclusion

At the very beginning of this paper, we listed the complexities involved in testing your website from different countries. We saw that it is mandatory to validate the availability, performance, and correctness of your website in order to ensure that your users are getting the best experience from your website no matter where they are located in the world.

The solution to complexity is to break problems down into smaller problem domains that are manageable. Test automation allows a web development team to systematically break down the problem of testing numerous versions of a website from numerous countries, into a manageable and maintainable effort. In this effort, Selenium, and other test automation frameworks, represent the path to success, in combination with a reliable global enterprise web proxy network. Building one initial test case from one country, done in a modular fashion, can easily be extended into more test cases and more countries. The rest is left as an exercise for the reader!

Questions & Answers

How do I know if a proxy is down, or if it's the website?

You're developers, so we know you need up-to-date information about what's happening on our network. If something stops working, you need to know if it's your app, your browser, or our proxy network. During any outage we'll try to keep you in the loop through Tumblr and our network status page.

WonderProxy provides an API (Application Programming Interface) that allows you to programmatically monitor WonderProxy's network status before you run your tests. In case the proxy server at the location you are about to test is down, you'll be able to set your test's failure reason accordingly or, if it suits your test plan, program your test to use an alternative location instead.

What is a CDN, and how can a proxy network help me test it?

> *A content delivery network (CDN) is a system of distributed servers (network) that deliver pages and other web content to a user, based on the geographic locations of the user, the origin of the webpage and the content delivery server.*[4]

A CDN caches the contents of your website in multiple locations and then serves that content from the location that is geographically closest to the end-user in order to make the content delivery time shorter. When we test our websites, we must test them from different geographical locations to ensure that the CDN we are using is working properly and delivers the correct content from the closest location to our users.

What about caching?

1. **Browser Caching** - each time a new web driver is launched from Selenium it has a clear cache, history, and cookies. There is no need to flush the cache programmatically before testing as long as you launch a new web driver instance (e.g., PhantomJS) for every test.

---

[4] *CDN - Content Delivery Network*. (n.d.). Webopedia. Retrieved March 8, 2020, from https://www.webopedia.com/TERM/C/CDN.html

2. **CDN caching** - CDN networks operate in a way that caches the contents of your website in different locations around the world (see the section on CDN networks). The caching process does not ensure completely that the end-user gets the latest content of your website but the latest content that has been cached by its hosts. You would want to ensure that your users, no matter where they are located and no matter by which CDN location they are served always get the most recent content of your website. In a case they aren't you should address this issue with your CDN provider.

How can localization testing help me use MRP software or my product database?

All ecommerce websites hold a product database with all their products. The case for most of these companies is that not all the products are available for purchase in all the countries where they operate, due to reasons such as shipping costs and local regulations.

While testing your ecommerce website it is important, yet challenging, to make sure you cover these regional exceptions to your product catalog. You'll have to access your website from different countries and extract from your product database the specific products that are not available for purchase at each of the countries you test (for example, make sure these products are marked on the website as unavailable, or confirm that they aren't displayed at all).

These are important tests that need to be conducted in order to ensure that your customers are able to purchase only the products you are able to deliver to them.

Of course, the easiest and most reliable way to perform such tests at scale, especially for ecommerce stores with many products and customers in different countries, is to automate them with a worldwide proxy network.