Selenium 4: New Feature - DevTools (APIs for CDP)

Java only

Ashis Raj

In this post, we will discuss one of the most promising features of Selenium 4 which is the new APIs for CDP (Chrome DevTools Protocol)! This addition to the framework provides a much greater control over the browser used for testing.

The new API allows you to send the built-in Selenium commands for CDP. These commands are wrapper methods that make it cleaner and easier to invoke CDP functions.

Selenium 4 has added native support for Chrome DevTools APIs. Chrome DevTools is a set of tools built directly into Chromium-based browsers like Chrome, Opera, and Microsoft Edge to help developers debug and investigate websites.

With Chrome DevTools, developers have deeper access to the website and are able to:

- Basic Authentication
- Intercept Network (mock the response of the request sent to a non-existent web page)
- Mock geolocations for location-aware testing (localization, internationalization, currencies, taxation rules, freight charges)
- Mock network speed (2G, 3G, 4G)
- Mock the device mode (dimensions) and exercise the responsiveness of the application
- Mock TimeZone
- Mock UserAgent
- Mock the APIs
- Capture and monitor the network traffic (Requests and Responses)

- Capture and monitor the performance
- Manipulate Cookies (delete, set and get cookies)
- Load insecure Web Pages
- Listen to JavaScript Exception
- Access/View console logs
- Add Custom Headers
- Block Resources (.css, .gif, .png, .jpg, .js, urls)
- Mutation observation (ability to capture DOM events)

and so much more…

Selenium 4 introduces the new ChromiumDriver class, which includes two methods to access Chrome DevTools: getDevTools() and executeCdpCommand().

The executeCdpCommand() method also allows you to execute CDP methods but in a more raw sense. It does not use the wrapper APIs but instead allows you to directly pass in a Chrome DevTools command and the parameters for that command. The executeCdpCommand() can be used if there isn't a Selenium wrapper API for the CDP command, or if you'd like to make the call in a different way than the Selenium APIs provide.

The Chromium-based drivers such as ChromeDriver and EdgeDriver now inherit from ChromiumDriver, so you also have access to the Selenium CDP APIs from these drivers as well.

Basic Authentication

If a website uses basic or digest authentication, it will prompt a dialog (browser popup) that cannot be handled through Selenium as it is only able to engage with DOM elements.

To go around that, it is possible to register an authentication method to access the content needed for the test.

Also, We can bypass this by using the CDP APIs to handle the authentication directly with DevTools.

Example

BasicAuthenticationViaAuthMethod.java (find it on my GithHub Repo.)
BasicAuthenticationViaCDPAPI.java (find it on my GithHub Repo.)

Intercept Network

Intercepting a network is applicable to scenarios where an end-to-end test is expected to run on a website that consists of a few non-existing pages, which are planned to be developed in the future.

Use this API to capture the network events of a website and intercept them as required.

Example

InterceptNetwork.java (find it on my GithHub Repo.)

Mock geolocations

Testing the location-based functionality of applications such as different offers, currencies, taxation rules, freight charges and date/time format for various geolocations is difficult because setting up the infrastructure for all of these physical geolocations is not a feasible solution.

With mocking the geolocation, we could cover all the aforementioned scenarios and more.

Example

MockGeoLocation.java (find it on my GithHub Repo.)

Mock Network Speed

Consider we have a scenario, we need to test our website how it loads under slow internet connection, like 2G, 3G, 4G, etc. If our website is tested under slow bandwidth conditions and if it is optimized to work in slow bandwidth it works normally.

The Selenium 4 with Chrome DevTools Protocol provides an option to control the network bandwidth it is called network emulation.

Example

MockNetworkSpeed.java (find it on my GithHub Repo.)

Mock the device mode (dimensions)

Most of the applications we build today are responsive to cater to the needs of the end users coming from a variety of platforms, devices like phones, tablets, wearable devices, desktops and orientations.

As testers, we might want to place our application in various dimensions to trigger the responsiveness of the application.

Example

MockDeviceModeViaCDPAPI.java (find it on my GithHub Repo.)
MockDeviceModeViaExecuteCdpCommand.java (find it on my GithHub Repo.)

Mock TimeZone

In case your website changes depending on your end-user's time zone, you can test that functionality by specifying a time zone you want to test in. You can configure your tests to run on a custom time zone.

Example

MockTimeZone.java (find it on my GithHub Repo.)

Mock UserAgent

Sometimes, we want to test mobile web apps and that's where UserAgent comes into play.

We can change our browser to a mobile browser by providing a mobile user agent.

Example

MockUserAgent.java (find it on my GithHub Repo.)

Mock the APIs

Consider an example, we have a website which internally calls some API, now what happens if we change the API parameters, what happens if we change the request information.

Mocking can be done in selenium using CDP commands.

Example

MockAPI.java (find it on my GithHub Repo.)

Capture and monitor the network traffic

We can capture the HTTP requests and responses of the application and access the url, method, data, headers, cookies, cache, status code and a lot more.

Example

CaptureHTTPRequest.java (find it on my GithHub Repo.)
CaptureHTTPResponse.java (find it on my GithHub Repo.)

Capture and monitor the performance

In today's fast world while we are iteratively building software at such a fast pace, we should aim to detect performance bottlenecks iteratively too. Poor performing websites and slower loading pages make unhappy customers.

Can we validate these metrics along with our functional regression on every build? Yes, we can!

Example

CapturePerformanceMetrics.java (find it on my GithHub Repo.)

Manipulate Cookies

We can clear browser cookies, delete all cookies, get and set cookie or cookies.

Example

ManipulateCookies.java (find it on my GithHub Repo.)

Load insecure Web Pages

The browser usually blocks loading the HTTPS website if the certificate has some errors. Many times, we will not be able to proceed with such website automation.

CDP provides the option to ignore those certificate errors and we can load that website on the browser and test it.

Example

LoadInsecureWebPages.java (find it on my GithHub Repo.)

Listen to JavaScript Exception

Exception, as commonly known, is an unusual event that breaks the normal execution flow of a program.

Use this API to listen to the JavaScript exceptions and register callbacks to process the exception details. These exception details can assist in further inspecting the cause and aiding the debugging process.

Example

ListenToJavaScriptException.java (find it on my GithHub Repo.)

Access Console/View Console Logs

We all rely on logs for debugging and analysing the failures. While testing and working on an application with specific data or specific conditions, logs help us in debugging and capturing the error messages, giving more insights that are published in the Console tab of the Chrome DevTools.

We can capture the console logs through our Selenium scripts by calling the CDP Log commands as demonstrated below.

Example

AccessConsoleLogs.java (find it on my GithHub Repo.)

Add Custom Headers

Adding extra header to the HTTP request is helpful when our application under test exposes filters requests or exposes specific features depending on the received headers.

Example

AddCustomHeaders.java (find it on my GithHub Repo.)

Block Resources

Page loading strategy: Defines the current session’s page loading strategy. By default, when Selenium WebDriver loads a page, it follows the normal pageLoadStrategy.

It is always recommended to stop downloading additional resources (like images, css, js), when the page loading takes a lot of time.

Consider an example, if you want to validate the website blocking all the .css files (that is website without styles), you need to block all the urls/files which are having extensions with .css.

Similarly, you might have a scenario to test blocking certain API calls.

Example

BlockURLAndResources.java (find it on my GithHub Repo.)

Mutation Observation

Mutation observation is the ability to capture DOM events you are interested in. For example, you might want to know if an element has changed its property value.

Before, the approach to this was to query the element continuously until the desired change occurred.

Now you can observe the changes in the DOM and assert over them when an incoming event notifies you about the expected change.

Example

ObserveChangesInDOM.java (find it on my GithHub Repo.)