**HEADLESS BROWSER**

A headless browser is **a browser simulation program that does not have a user interface**. One of the significant benefits of using headless browsers is performance. Since headless browsers don't have a GUI, they are faster than real browsers.

This allows a headless browser to deliver the real browser perspective without demanding memory or cost incurred test cases with actual browser GUI.

**WHY TO USE HEADLESS BROWSER** ?

The reason behind using a headless browser or carrying out headless testing is that **they are faster than the normal browsers**. Since you are not opening the browser GUI, you can ignore the time taken by a normal browser to load CSS, JavaScript and render HTML

Top Headless Browsers Available

## Puppeteer

Puppeteer is a Node.js library that provides a high-level API to control headless Chrome or Chromium developed by the chrome team.

## HTML Unit

HTML Unit is a headless browser written in java code. It has JavaScript support that handles complex Ajax libraries. It supports the Cookies, HTML responses (clicking on hyperlinks, submitting forms, etc.), HTTPS security and basic HTTP authentication.

## Headless Firefox

For Firefox 56 or above versions, Mozilla Firefox offers headless testing. Instead of using some other tools to simulate browser environments, developers can now connect with inbuilt APIs provided within Mozilla to run Firefox in headless mode and test a variety of use cases.

## Headless Chrome

Chrome version 59 or later supports headless mode. It carries all modern web platform features contributed by Chromium and the Blink rendering engine to the command line.

## PhantomJS

PhantomJS is a headless browser scriptable with JavaScript API based on WebKit. It can be used for, Screenshots, Page Automation, Network Monitoring, to render dashboard screenshots for users, to run Unit tests on the command line and to gather the information from the web.

## TrifleJS

TrifleJs is a headless Internet Explorer browser that uses the V8 Javascript engine to run the test scripts.

# When To use Headless Testing?

Here are some of the scenarios where headless browser testing has to be preferred:

* Automate the HTML responses or web interactions like mouse-clicking, form submission, simulate keyboard input, etc.
* Handle Ajax calls and JavaScript execution.
* Generates screenshots and PDFs of webpages.
* Scrapes content and extracts the data from websites
* Monitors the Network
* No GUI hence less resource overhead.
* Since it interprets HTML and CSS like a real browser, it can be used to test the style of the elements such as page layouts like width and height of the page, ensuring the coordinates of an element or checking the font style or color etc.
* Simulating multiple browsers on a single machine without the resource overhead.

# When Not To Consider Headless Browser Testing

* Want to test as end-user
* The need to observe the test visually
* Debugging the browser in headless mode is quite complex

# Conclusion

Headless browser testing is very helpful to developers before they commit their code so that they get the feedback quickly to give rapid fix at the early stage. Of course, Headless testing is not a universal testing solution. If you want to test the application visually you need a real browser. Headless Browser testing plays a major role when performance and time are crucial.

PROGRAM :----

from selenium import webdriver

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.common.by import By

from webdriver\_manager.chrome import ChromeDriverManager

"""Two types we can create headless browsers"""

#TYPE 1

opt = Options()

opt.add\_argument("--headless")

driver = webdriver.Chrome(ChromeDriverManager().install(),chrome\_options=opt)

driver.get("http://www.google.com")

driver.get\_screenshot\_as\_file("headlessscreenshot.png")

#TYPE 2

opt = Options()

opt.headless=True

#by default it will be false

driver = webdriver.Chrome(ChromeDriverManager().install(),chrome\_options=opt)

driver.get("http://www.google.com")

driver.get\_screenshot\_as\_file("headlessscreenshot.png")