Headless Browser in Selenium

Selenium is one of the most popular and efficient browser automation tools available today. With Selenium Web Driver, it automates several (browser) actions, such as opening a webpage, clicking a link, checking the URL, and so on.

But there are cases when you may need to run automation tests in 'headless' mode, i.e., when no browser is being displayed. In these cases, you can execute Selenium tests in headless browsers.

What Is A Headless Browser?

A headless browser is a web-browser **without a graphical user interface**. This program will behave just like a browser but will not show any GUI.

A headless browser is a browser simulation program that does not have a user interface. These programs operate like any other browser, but do not display any UI. When Selenium tests are run, it executes in the background.

While there are several such headless browsers available in the market, the following are the most popular ones:

- Headless Chrome
- PhantomJS
- SlimerJS
- TrifleJS
- HTMLUnit driver
- ZombieJS
- Watir-webdriver

Features of HTML unit driver

- Support for the HTTPS and HTTP protocols
- Support for HTML responses (clicking links, submitting forms, walking the DOM model of the HTML document etc.)
- Support for cookies
- Proxy server support
- Support for basic and NTLM authentication
- Excellent JavaScript support
- Support for submit methods GET and POST
- Ability to customize the request headers being sent to the server
- Ability to determine whether failing responses from the server should throw exceptions or should be returned as pages of the appropriate type

Useful Links

https://gist.github.com/evandrix/3694955
You can find the official https://sourceforge.net/projects/htmlunit/files/htmlunit/
https://htmlunit.sourceforge.io/gettingStarted.html

Three Benefits Of Headless Browser Testing

1. Improves speed and performance
Since this type of testing does not actually open a browser, the system saves the processing power that would otherwise be used in a real browser test. Consequently, the tests are executed faster.
2. Allows testing browserless setups
There may be setups where installing a browser is not possible, such as servers. In these cases, headless browsers help run automation tests easily.
3. Helps you multitask
You can use your browser or your machine to do anything else while the tests run in the background. Save hours of time that is otherwise spent staring at the screen.

How To Run Selenium Tests In Headless Google Chrome

Google recently introduced a headless option for Chrome. It is available from version 59. So, if you are planning to run headless tests, you need Google Chrome browser with version 59 or above.

Before you can begin testing, there are a couple of things you'll need to install first:

- Java
- Selenium
- ChromeDriver (latest)

ChromeOptions is a class in Selenium to set the arguments to ChromeDriver. In the following example, we will pass the two arguments to ChromeDriver to run in headless mode.

```
package ChromeOptions;

import java.io.File;
import java.io.IOException;

import org.apache.commons.io.FileUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

public class HeadlessBrowserDemo {
        public static void main(String[] args) throws IOException {
                System.setProperty("webdriver.chrome.driver",
                                "D:\\Selenium\\Selenium Browsers Jars\\Chrome 84\\chromedriver.exe");
                ChromeOptions options = new ChromeOptions();
                options.addArguments("headless");
                options.addArguments("window-size=1366x768");
                WebDriver driver = new ChromeDriver(options);
                // driver.get("https://contentstack.built.io");
                driver.get("https://www.google.co.in/");

                System.out.println("Browser Title is: " + driver.getTitle());

                File scrFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
                FileUtils.copyFile(scrFile, new File("./ScreenShot/Demo.png"));
                driver.quit();
        }}
```

PhantomJS

PhantomJS is a headless browser with JavaScript API. It is an optimal solution for Headless Website Testing, access and manipulate webpages & comes with the standard DOM API.

In order to use PhantomJS with Seleniun, one has to use GhostDriver. **GhostDriver** is a implementation of Webdriver Wire protocol in simple JS for PhantomJS.

The latest release of PhatomJS has **integrated** GhostDriver and **there is no need to separately install it.**

Steps to run Selenium with PhatomJS

Step 1) You need Eclipse with Selenium installed
Step 2) Download PhantomJS here

https://phantomjs.org/download.html

```
package phantomJsDemo;

import java.io.File;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.phantomjs.PhantomJSDriver;

public class LoginTest {
	public static void main(String[] args) {
		File file = new File("D:\\Selenium\\phantomjs-2.1.1-windows\\bin\\phantomjs.exe");
		System.setProperty("phantomjs.binary.path", file.getAbsolutePath());
		WebDriver driver = new PhantomJSDriver();
		driver.get("http://www.google.com");
		WebElement element = driver.findElement(By.name("q"));
		element.sendKeys("Bangalore");
		element.submit();
		System.out.println("Page title is: " + driver.getTitle());
		driver.quit();
	}
}
```

Many organization uses Phantom.JS for various purpose, for example,

- Headless Testing
- Screen Capture
- Page Automation
- Network Monitoring
- To render dashboard screenshots for their users
- To run Unit tests on command line
- To generate employee handbooks from HTML to PDF
- Combined with QUnit for the test suite

Advantage and Disadvantage of headless browsers or Why should I use this?

Ans-One of the most Important advantage is Performance.

1-When we run your test case using headless browsers then you will get the result just in seconds, you will see the result in below program.

2-When we have to run a test case to create some sample test data or just you have to verify some messages and functionality then you can try headless browsers.

3- When we have to run the test case on the remote machine or server, which does not have any browser, but still you have to execute test case then you can try with headless browsers again.

When you build your test case using Jenkins then also it runs in Headless mode.