**Selenium Headless Browser Testing**

Web browsers are an integral part of the UI automation using SeleniumWebDriver. Launching a browser and then executing the test cases on the browsers are an essential part of the web automation testing. But when we run the Selenium tests on any of the browsers, we generally face some challenges such as slow rendering on the browser, interference of other applications running on the system, etc. Apart from these, the major of the [***CI***](https://codeship.com/continuous-integration-essentials#:~:text=Continuous%20Integration%20(CI)%20is%20a,CI%20it%20is%20typically%20implied.) systems these days are Non-UI (such as Unix based systems). Therefore, for running the test cases on those systems, we need a way to run the test cases in a Non-UI mode, and this is where the headless browser comes into the picture, which helps in the execution of the Selenium Headless Browser tests in a Non-UI mode.

Almost all modern browsers provide the capability to run them in headless mode. In this article, we will discuss these headless browsers and how we can use them for running the Selenium test cases in headless mode.

* What is a headless browser?
  + Why use a headless browser for test execution?
  + Limitations of headless testing.
* How to run Selenium tests in headless mode?
  + Running Selenium headless browser tests using the HTMLUnitDriver.
  + Running Selenium headless browser tests using the headless Chrome browser.
  + And running Selenium headless browser tests using the headless Firefox browser.
  + Running Selenium headless browser test cases using a headless Edge browser.

## What is a headless browser?

A headless browser is a term used to define browser simulation programs that do not have a GUI. These programs execute like any other browser but do not display any UI. In headless browsers, when Selenium tests run, they execute in the background. Almost all modern browsers provide the capabilities to run them in a headless mode.

So, is headless testing possible using [***Selenium***](https://www.toolsqa.com/selenium-webdriver/selenium-testing/)? Yes, Selenium supports headless testing. In older versions of Selenium, we used the HTMLUnitDriver mainly, a headless driver providing a Non-GUI implementation of Selenium WebDriver. But with the latest versions of SeleniumWebDriver 3 and SeleniumWebDdriver 4, Selenium also supports headless versions of real browsers like Chrome, Firefox, and Edge. Selenium provides various configurations using which we can run these browsers in a headless mode. Before understanding the details of these configurations, let's first understand what benefits does running a test in headless mode provides?

### ***Why use a headless browser for test execution?***

Running a Selenium test case in headless mode provides the following benefits:

* ***Useful in CI pipeline:*** When we need to execute automated test cases remotely on a server or in any of the build and release pipelines for ***continuous integration*** servers like [***Jenkins***](https://www.jenkins.io/), it is not always possible to install real browsers on such remote machines. We can use headless browsers to run automation tests efficiently.
* ***Beneficial in web scraping:*** When you want to write a web scraper or data extractor that needs to visit some websites and collect data, headless browsers are a perfect choice. Because we are not concerned about functionality in these cases, we visit web pages and get the data.
* ***Support for multiple browser versions:*** Sometimes, the tester would like to simulate multiple browser versions on the same machine. In that case, you would want to use a headless browser because most of them support the simulation of different versions of browsers.
* ***Faster automation test execution:*** The performance with a headless browser is better compared to real browser automation. The real browsers like Google Chrome, Firefox, and Internet Explorer take a significant amount of time to load ***CSS, JavaScript,*** Images, and open and render ***HTML.*** Headless browsers do not require all this to load and will start performing functions without waiting for a page to load completely. When we need to run the ***regression*** scripts, in headless browsers, we could save time as there are much faster and can render results quickly.
* ***Multi-Tasking:*** Headless browsers can help you, multi-task.You can use your browser or your machine to do anything else while the tests run in the background. Save hours that we otherwise spend staring at the screen.

Even though running the Selenium test provides these benefits, certain limitations are also attached to the same. A few of them are as listed below:

### ***Limitations of headless browser testing***

* Debugging will not be feasible, as the only way to check what's running on the browser is to grab the screenshots and validate the output.
* Headless browsers don't mimic the exact user behavior, as the page doesn't render precisely with all the dependencies that it will render in an actual browser.
* Cosmetic bugs like the location of a web element, the color of a web element may get missed while running the tests in headless mode.

Let's now see how we can run the Selenium test cases in a headless mode:

## How to run Selenium test cases in headless mode?

As we discussed above, Selenium WebDriver provides various drivers and configurations which help in running the Selenium test cases in headless mode. Let's understand a few of them in the following sections:

* Running Selenium test cases using HTMLUnitDriver.
* Running Selenium test cases using a headless Chrome browser.
* And, Running Selenium test cases using the headless Firefox browser.
* Running Selenium test cases using a headless Edge browser.

### ***Running Selenium headless browser tests using HTMLUnitDriver.***

***HtmlUnitDriver*** is an implementation of Selenium WebDriver based on ***HtmlUnit,*** which is a Java-based implementation of a web browser without a GUI. HtmlUnitDriver is currently the fastest and most lightweight implementation of WebDriver. HtmlUnitDriver is a well known headless browser driver and has many advantages -

* Fastest implementation of WebDriver compared to other browsers.
* HtmlUnitDriver is platform-independent.
* HtmlUnitDriver supports JavaScript.
* Also, HtmlUnitDriver allows you to choose other browser versions to run your scripts. You can mention different browser versions of Chrome or Firefox in the HtmlUnitDriver itself.

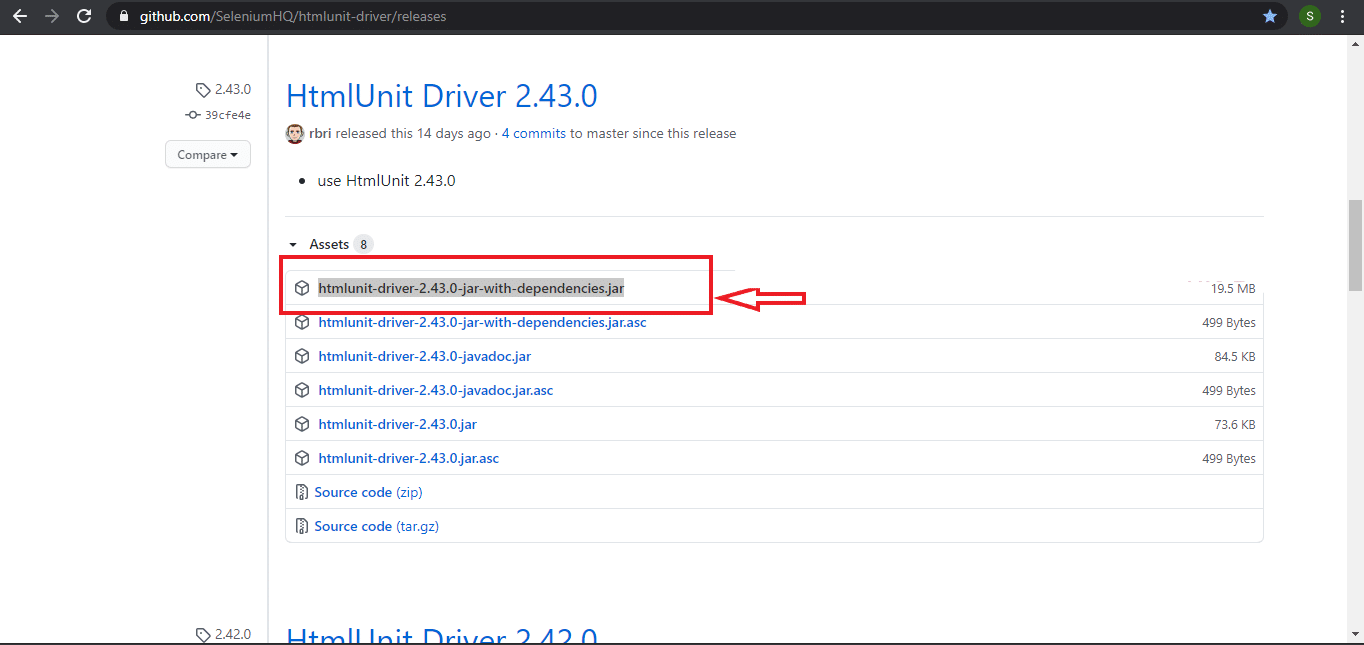
#### **How to use HtmlUnitDriver as a headless browser with Selenium?**

To implement headless testing, Selenium makes use of HtmlUnitDriver, which is another implementation of WebDriver, similar to FirefoxDriver, ChromeDriver, etc. HTMLUnitDriver is available as an external dependency and requires you to add the library explicitly.

***Note:*** In the earlier versions of Selenium (before version 2.53), HtmlUnitDriver was available within the selenium library, and there was no need to download jar externally. While this driver is still supported, it is now a separate dependency and uses the Html Unit framework. These dependencies need to add in the project in the same way as other jars add.

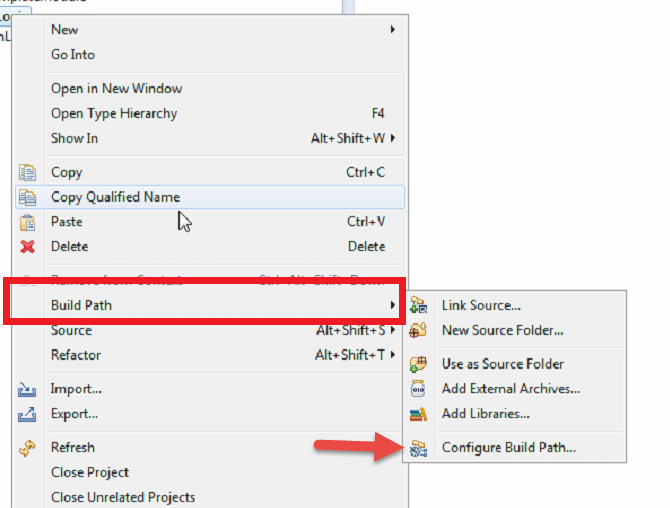
To download the HtmlUnitDriver dependencies, follow the steps as mentioned below:

1. Navigate to [***https://github.com/SeleniumHQ/htmlunit-driver/releases***](https://github.com/SeleniumHQ/htmlunit-driver/releases).
2. Click on the latest version of the HTML unit driver as shown in the image below:

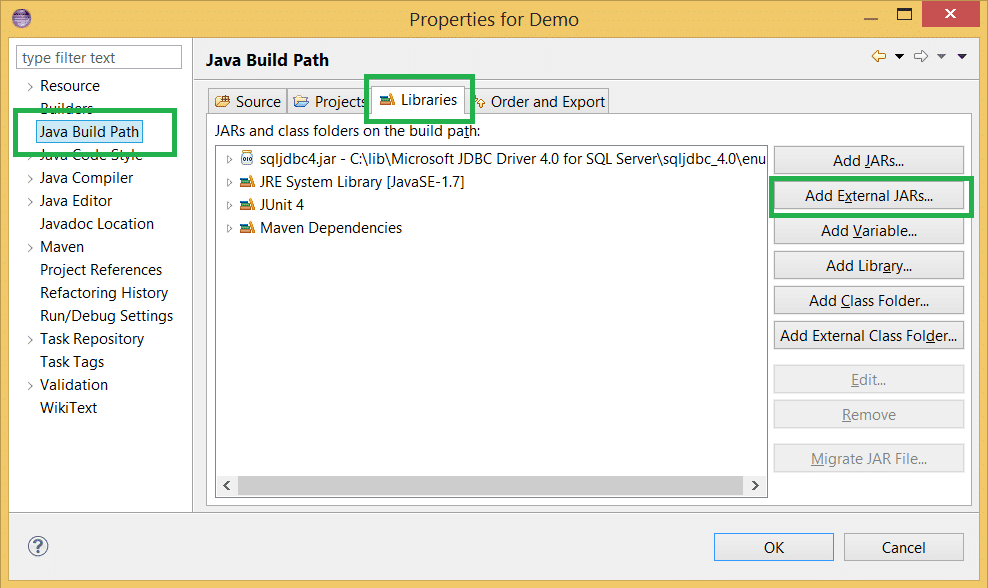


After downloading the jar file, follow the below steps to add the jar to the [***Eclipse***](https://www.toolsqa.com/selenium-webdriver/configure-selenium-webdriver-with-eclipse/) project,

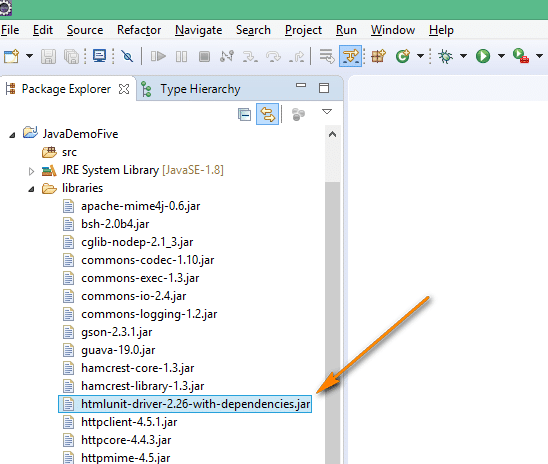
1. Right-click on your project.
2. Select ***Build Path***  and Click on ***Configure Build Path***



1. Click on the ***Libraries*** tab and select ***Add External JARs.***



1. Select the downloaded jar file from the folder.
2. Click and Apply and Ok. The jar will add as below -



Once you have the jar added to the Eclipse project, you can import the class "***org.openqa.selenium.htmlunit.HtmlUnitDriver***" into your test code by adding the below line -

import org.openqa.selenium.htmlunit.HtmlUnitDriver;

After that, You can create a HtmlUnitWebDriver instance as below -

HtmlUnitDriver unitDriver = new HtmlUnitDriver();

The above syntax will construct a new instance with [***JavaScript***](https://www.toolsqa.com/javascript/javascript-tutorial/) disabled, and the default browser version. If your application needs some of the JavaScript functionalities, you can enable the same, with the configurations as mentioned in the following article:

#### **Enabling JavaScript support in HtmlUnitWebDriver**

HtmlUnitWebDriver, by default, doesn't enable [***JavaScript***](https://www.toolsqa.com/javascript/javascript-tutorial/). It means in the above code, if you try to execute JavaScript, an error will be thrown, stating that JavaScript is not enabled. To create a driver instance with JavaScript enabled, you can use the constructor ***HtmlUnitDriver*** (boolean enable javascript) and set the value as true as shown below

HtmlUnitDriver unitDriver=HtmlUnitDriver(true)

So, this will create an instance of the HtmlUnitDriver, with JavaScript enabled.

Now that we have created the instance of the WebDriver let's see how we can use the same in our Selenium test scripts for running the tests in a headless mode.

#### **Example showing usage of HTMLUnitDriver to run Selenium test cases in Headless mode:**

Suppose we need to automate and run the following test scenario using Selenium HTMLUnitDriver:

1. Launch the website with the URL - [***"https://demoqa.com/"***](https://demoqa.com/).
2. Print the Title of the page.

We can achieve the same using the following code snippet:

import org.openqa.selenium.htmlunit.HtmlUnitDriver;

public class headlessBrowserDemo {

public static void main(String[] args) {

// Declaring and initialising the HtmlUnitWebDriver

HtmlUnitDriver unitDriver = new HtmlUnitDriver();

// open demo site webpage

unitDriver.get("https://demoqa.com/");

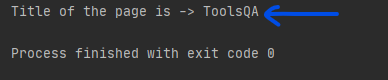
//Print the title of the page

System.out.println("Title of the page is -> " + unitDriver.getTitle());

}

}

When you run the code, you will not see any browser on the screen, but as the execution completes, it will print the title of the page as shown below -



As highlighted in the above image, the title of the page prints as ***"ToolsQA"***.

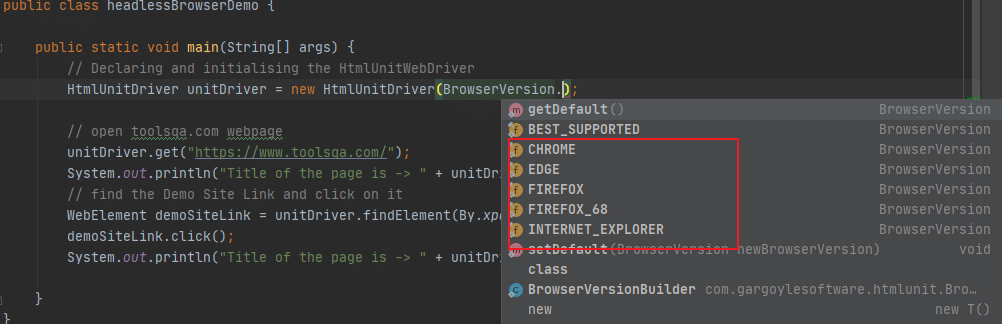
Now, when we ran the test cases with Default options of HTMLUnitDriver, it picked the browser. This browser was by default on your machine. If you want to execute the test cases in a specific browser version also, you can also do the same with the help of HTMLUnitDriver. Let's understand how we can achieve the same:

#### **How to run tests on different browser versions using HtmlUnitDriver?**

HtmlUnitDriver allows you to select the version of the browser that you would like to run your tests. The main supported browsers are ***Chrome, Firefox, and Internet Explorer.*** We can use the different versions of the browser by passing the values in the constructor that will create a new instance with the specified version of the browser as in the code below -

HTMLUnitDriver driver = new HTMLUnitDriver(BrowserVersion.Firefox\_68);

You can use different browser versions as shown in the image below-



Also, if you want to enable JavaScript in a particular browser version, there is another constructor for HTMLUnitDriver that allows creating a new instance with the specified Browser Version and the JavaScript support by setting the value = true as in the below line of code -

HtmlUnitDriver unitDriver = new HtmlUnitDriver(BrowserVersion.FIREFOX\_68,true);

Let's try to write a complete code using HTMLUnitDriver using Firefox version 68 -

import com.gargoylesoftware.htmlunit.BrowserVersion;

import org.openqa.selenium.htmlunit.HtmlUnitDriver;

public class headlessBrowserDemo {

public static void main(String[] args) {

// Declaring and initialising the HtmlUnitWebDriver using Firefox 68 version

HtmlUnitDriver unitDriver = new HtmlUnitDriver(BrowserVersion.FIREFOX\_68);

// open demo site

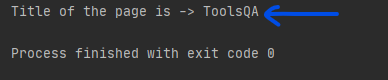
unitDriver.get("https://www.demoqa.com/");

System.out.println("Title of the page is -> " + unitDriver.getTitle());

}

}

Again, you will not see any browser on the screen when the above code executes, and the page title will print in the output:



As highlighted in the above image, the title of the page prints as ***"ToolsQA"***.

### ***Running Selenium headless browser tests using the headless Chrome browser.***

[***Headless Chrome***](https://developers.google.com/web/updates/2017/04/headless-chrome) is a way to run the Chrome browser in a headless environment without the full browser user interface. Headless Chrome offers you a real browser context without the memory overhead of running a full version of Chrome. Google Chrome is available with headless execution since version 59. Selenium WebDriver provides a class called ***"ChromeOptions"***, which can specify certain configurations to change the default behavior of Chrome. One of those configurations is the ***"headless"*** mode, which launches the Chrome in headless mode while running the test cases. The following code snippet shows how we can pass the ***"headless"*** option using the ChromeOptions class. It also displays how we can then use those options while creating the instance of the ChromeDriver:

ChromeOptions options = new ChromeOptions()

options.addArgument("headless");

ChromeDriver driver = new ChromeDriver(options);

In the above code, the browser is instructed to run in the headless mode using the ***addArgument()*** method of the ChromeOptions class provided by the Selenium WebDriver.

Now, suppose we have to run the same test case of the previous section, where we run the Selenium test using HTMLUnitDriver, on the ***Chrome headless mode,*** we can modify the code as shown below:

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

public class headlessChromeDemo {

public static void main(String[] args) {

//declare the chrome driver from the local machine location

System.setProperty("webdriver.chrome.driver", "C:\\chromedriver.exe");

//create object of chrome options

ChromeOptions options = new ChromeOptions();

//add the headless argument

options.addArguments("headless");

//pass the options parameter in the Chrome driver declaration

WebDriver driver = new ChromeDriver(options);

//Navigate to toolsQA site url

driver.get("https://demoqa.com/");

//Print the Title of the Page

System.out.println("Title of the page is -> " + driver.getTitle());

//Close the driver

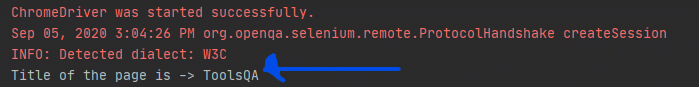
driver.close();

}

}

***Note:*** For setting up ChromeDriver and running the Selenium tests on the Chrome browser, you can refer to the article [***"Running tests in Chrome browser"***](https://www.toolsqa.com/selenium-webdriver/run-selenium-tests-on-chrome/).

The output/ result of the above code will print the title of the page.



The above screenshot highlights the title of the page that prints using the code.

### ***Running Selenium headless browser tests using the headless Firefox browser.***

For ***Headless Firefox,*** we have to work in the same way as we did in headless Chrome. First, we will set the path for the [***Gecko Driver***](https://www.toolsqa.com/selenium-webdriver/selenium-geckodriver/) (used for Firefox browser) and then set the options as headless. Here, we use the methods ***setHeadless*** (true) of the ***FirfoxOptions*** class provided by Selenium WebDriver.

You can use the below code to use ***Firefox*** in ***Headless*** mode -

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.firefox.FirefoxOptions;

public class headlessFirefoxDemo {

public static void main(String[] args) {

//set the path of the Gecko driver as per the location on local machine

System.setProperty("webdriver.gecko.driver", "C:\\geckodriver.exe");

//Set Firefox Headless mode as TRUE

FirefoxOptions options = new FirefoxOptions();

options.setHeadless(true);

//pass the options parameter in the Firefox driver declaration

WebDriver driver = new FirefoxDriver(options);

//Navigate to toolsQA site url

driver.get("https://demoqa.com/");

//Print the Title of the Page

System.out.println("Title of the page is -> " + driver.getTitle());

//Close the driver

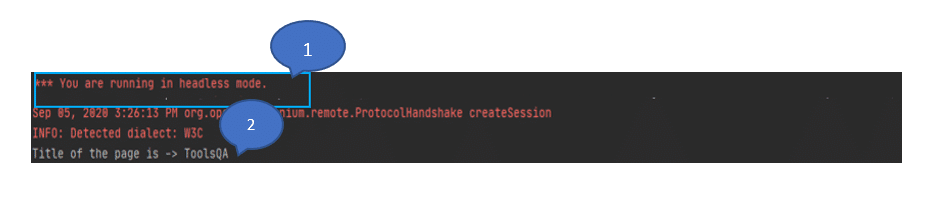
driver.close();

}

}

For setting up FirefoxDriver and running the Selenium tests on the Firefox browser, you can refer to the article [***"Running tests in Firefox browser"***](https://www.toolsqa.com/selenium-webdriver/selenium-geckodriver/).

The output/ result of the above code will be as below -



In the above screenshot, you can see that the script execution clearly states, ***"Headless Mode is running"***, as highlighted by marker '1', and then the title of the page is printed as 'ToolsQA' as highlighted by marker '2 '.

### ***Running Selenium headless browser tests using the headless Edge browser.***

For headless execution in the ***Edge*** browser, we will use ***EdgeOptions*** Class to configure the headless execution. The ***EdgeOptions*** is a class to manage options specific to Microsoft Edge Driver. You can configure the settings for Edge Driver using this Class. It is similar to what we did for Chrome Browser using ChromeOptions in the above topic in this article. As the basis of the Edge is majorly the Chromium platform, so its options will also be almost similar to Chrome. You can use the following code snippet to set the ***"headless"*** option of the EdgeDriver.

EdgeOptions edgeOptions =new EdgeOptions();

edgeOptions.addArguments("headless");

WebDriver driver= new EdgeDriver(edgeOptions);

To run Edge in headless mode, you have to use the ***addArguments()*** method. And, pass the value as ***"headless"*** to instruct the driver for headless testing.

We can modify the above tests to run it on ***Edge headless,*** as shown below:

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.edge.EdgeDriver;

import org.openqa.selenium.edge.EdgeOptions;

public class headlessEdgeDemo {

public static void main(String[] args) {

// Set the edge driver path as per the machine location

System.setProperty("webdriver.edge.driver","C:\\seleniumDemo\\msedgedriver85.exe");

//Initialize the EdgeOptions class

EdgeOptions edgeOptions =new EdgeOptions();

//Use the addArguments method for configuring headless

edgeOptions.addArguments("headless");

//Pass the edgeOptions object to the Edge Driver

WebDriver driver= new EdgeDriver(edgeOptions);

//Navigate to toolsQA site url

driver.get("https://demoqa.com/");

//Print the Title of the Page

System.out.println("Title of the page is -> " + driver.getTitle());

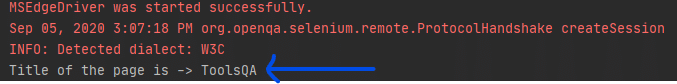
//Close the driver

driver.close();

}

}

The output/ result of the above code will be as follows:



***Note:*** Earlier, there were other headless browsers like PhantomJS, Opera, which we used for headless testing, but now we can use Headless Chrome or Firefox, which can serve most of the headless testing. As per the latest updates from Selenium, with the Selenium 4 Alpha release, support for ***Opera*** and ***PhantomJS*** browsers has been removed. That's because the WebDriver implementations for these browsers are no longer under active development.