Headless Testing: Executing the web applications' UI tests without opening a browser’s user interface is called headless browser testing. Headless browser acts similar to a normal web browser.

**When to use headless browser testing:**

1. We can use headless testing once the cross browser testing is completed and want to run regression test cases in subsequent releases and with continuous integration
2. You have no option other than using headless testing when your machine does not have a GUI, for instance if you want to run your tests in unix.
3. It is recommended to use headless browser when tests are executed in parallel as User Interface based browsers consumes a lot of Memory / resources. So headless browsers can be used for server side performance testing too.

**Disadvantages of Headless browser testing:**

1. Hard to debug inconsistent failures on locating elements due to page loading too fast
2. In Real browser as functions are performing in front of user and he can interact with it so he can easily detect where the tests goes fail. And can easily debug if anything goes wrong.
3. Unintended interactions (losing the benefit of automated UI testing vs integration or unit testing) due to speed/headless state of browser.
4. Headless browsers aren’t representing real users, as no user uses the your application without UI. Because it doesnot have UI, it may not report errors related with images.

**Selenium support for headless browser**

Selenium supports headless testing using its class called**HtmlUnitDriver.**This class internally uses HtmlUnit headless browser. HtmlUnit is a pure Java implementation so you will not find this tutorial to be focused on java bindings **Quick start !** You can create a HtmlUnitWebDriver like this

**package** seleniumTests;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.chrome.ChromeOptions;

**import** org.openqa.selenium.htmlunit.HtmlUnitDriver;

**import** factory.DataProviderFactory;

**public** **class** HeadlessChrome {

**public** **static** **void** main(String[] args) {

HtmlUnitDriver unitDriver=**new** HtmlUnitDriver();

unitDriver.get("https://www.freecrm.com/index.html");

unitDriver.findElement(By.*name*("username"));

unitDriver.findElement(By.*name*("password"));

System.***out***.println("entered username and password");

unitDriver.findElement(By.*xpath*("//input[@type='submit']"));

System.***out***.println("Clicked on Submit button");

unitDriver.close();

}

}

Javascript support in HtmlUnit WebDriver

HtmlUnitWebDriver by default doesn’t does not enable JavaScript. This means in the above code if you try to execute java script and error will be thrown stating that JavaScript is not enabled. To enable Java script in two ways

1)By passing the java script enable flag in constructor

**HtmlUnitDriver unitDriver = new HtmlUnitDriver(true);**

2)By specifying it using the **setJavascriptEnabled**method

**HtmlUnitDriver unitDriver = new HtmlUnitDriver();**

**unitDriver.setJavascriptEnabled(true);**

Once you have html unit webdriver set for Java script you can execute any script exactly like you would do in any other browser object. Here is a sample code.

HtmlUnitDriver unitDriver = new HtmlUnitDriver();

unitDriver.setJavascriptEnabled(true);

unitDriver.get("http://google.com");

String domainName = (String) unitDriver.executeScript("return document.domain");

System.out.println("Domain name is " + domainName);

**Testing on Different Browser versions.**