|  |
| --- |
|  |
| What is WebDriver: |
| WebDriver is an interface that is version 2.0 and 3.0 of selenium. |
| Name driver implementation for different browsers? |
| • FirefoxDriver  • ChromeDriver  • InternetExplorerDriver  • SafariDriver  • EventFiringWebDriver  • HTMLUnitDriver  • RemoteWebDriver |
| How to Instantiate webdriver? |
| WebDriver driver = new FirefoxDriver(); |
| How to open the web application? |
| driver.get(“http://www.google.com”) |
| After what Firefox version, Firepath is not working? |
| Firefox -v 56 |
| Which tool you use to find XPath on Firefox now? |
| 1. Firefox dev tool:      1. Right Click on the element > Copy > Xpath. 2. Try XPath |
| Which tool you use to find XPath on Chrome now? |
| 1. Chrome dev tool : 2. inspect element > Ctrl+f        1. Right Click on the element > Copy > Copy Xpath. 2. Sel Assist      1. ChroPath:   Right Click on element > Inspect > on R.H.S window, you will see ChroPath option. You can write your own Xpath or click on Dom elements(ChroPath will auto generate XPath for it).     1. Ranorex Selocity:   Open Ranorex Selocity from browser.  User can use any of the below way to generate/validate the xpath:   1. Right Click on the element > Choose ‘Copy Selector’ option of Ranorex > xpath 2. Click on any dom element, xpath will be auto generate. 3. Write your own xpath |
| How can we create a basic Selenium WebDriver project? |
| 1. Download “Selenium Client & WebDriver Language Bindings” from <http://www.seleniumhq.org/download/>      1. Create a Java project      1. Right Click on Project (“Basic”) > Build Path > Configure Build Path > Under Library tab > Add External Jars   Add all the Client & WebDriver Language Bindings jar files > Click Ok xtrac     1. Now create a package and class under the project, here you can use webdriver interface. 2. Download Browser driver example for FF it’s available below: <https://github.com/mozilla/geckodriver/releases> 3. Extract the driver .exe file.      1. Write below code and provide FF browser driver exe path : |
| How to set driver path for IE, Firefox and Chrome ? |
| Way1 : Using driver path using : System.setProperty    Way2 : Set driver’s directory to system path and you will not need “System.setProperty” statement: |
| What are the setting you should keep for IE to work without much issues? |
| 1. Keep, protected setting same (either enable or disable) for all zones (internet, Local intranet, trusted sites, restricted sites).      1. Set the below capabilities: |
| How to run Selenium webdriver on Safari? |
| Important thing to notice:  Safari Automation is only possible on Mac, it is not supported on Windows by Apple and Selenium WebDriver Automation  Please do these steps:  1. Download the "Selenium Standalone Server" file from <http://www.seleniumhq.org/download/>  2. Right click and open the file, confirm by clicking Open button, please see the screenshot.     1. Open Develop menu in Safari:     4. Click on Develop menu -> Select -> Allow Remote Automation    Note: Before Selenium WebDriver 3.x , we have to download safari extension and enable it. now it is not needed after Selenium WebDriver 3.x version.   1. Now we can run below code(no need to set SystemPath as well ) : |
| List some of the Locators ? |
| 1. Id : Select element with the specified @id attribute. 2. Name Select first element with the specified @name attribute. 3. Linktext Select link (anchor tag) element which contains text matching the specified link text 4. Partial Linktext Select link (anchor tag) element which contains text matching the specified partial link text 5. Tag Name Locate Element using a Tag Name . 6. Class name Locate Element using a class Name .. 7. Css Select the element using css selectors. 8. Xpath Locate an element using an XPath expression. |
| How to find element using id locator? |
| WebElement searchbox= driver.findElement(By.id("….")); |
| How to find element using name locator? |
| WebElement searchbox= driver.findElement(By.name("….")); |
| How to find element using LinkText locator? |
| WebElement loginLink= driver.findElement(By.linkText("…..")); |
| How to find element using Practial LinkText locator? |
| WebElement loginLink= driver.findElement(By. partialLinkText ("…..")); |
| How to find element using Class locator? |
| WebEment theme = driver.findElement(By.className("….")); |
| How to find element using TagName locator? |
| List<WebElement> myElementListByTag = driver.findElements(By.tagName("…")); |
| How to use CSS (Cascading Style Sheets)? |
| Syntax:  tag[attribute<special character>=’value’]   |  | | --- | | Alternative Symbols of id:   * “#” -> Id * Example: input[id=displayed-text] can be written as  1. input#displayed-text 2. #displayed-text | | Alternative Symbols of class:   * “.” -> Class * Example: input[class=displayed-class] can be written as  1. input.displayed-class 2. .displayed-class  * We can also write : .class1.class2.class3 -> Until we find a unique element * .inputs.displayed-class | | Using special character/wildcards in CSS Selectors:   1. “^” -> Represents the starting text 2. “$” -> Represents the ending text 3. “\*” -> Represents the text contained   Examples:  input[class^='inputs'] -> Two matching nodes  input[class$='class'] -> One matching node  input[class='displayed-class'] - No matching nodes  input[class\*='displayed-class'] -> One matching node | | Finding Children: using ‘>’ symbol.  Examples:  fieldset>table  fieldset>#product -> One matching node  fieldset>button -> One matching node  fieldset>a  fieldset>input#name |  |  | | --- | |  | | Alternative Symbols of id:   * “#” -> Id * Example: input[id=displayed-text] can be written as  1. input#displayed-text 2. #displayed-text | | Alternative Symbols of class:   * “.” -> Class * Example: input[class=displayed-class] can be written as      1. input.displayed-class 2. .displayed-class  * We can also write : .class1.class2.class3 -> Until we find a unique element * .inputs.displayed-class | | Using special character/wildcards in CSS Selectors:   1. “^” -> Represents the starting text 2. “$” -> Represents the ending text 3. “\*” -> Represents the text contained   Examples:    input[class^='inputs'] -> Two matching nodes  input[class$='class'] -> One matching node    input[class='displayed-class'] - No matching nodes  input[class\*='displayed-class'] -> One matching node | | Finding Children: using ‘>’ symbol.  Examples:    fieldset>table  fieldset>#product -> One matching node    fieldset>button -> One matching node    fieldset>a    fieldset>input#name | |
| How to use XPath? |
| Syntax:  //tag[@attribute='value']       |  | | --- | | https://letskodeit.teachable.com/p/practice | | //div[@id='navbar']/div/div/div/ul/li[2]/a | | .//div[@id='navbar']//ul/li[2]/a[@class='navbar-link'] | | //div[@class='homepage-hero']//a[text()='Enroll now'] | | //div[@class='navbar-header']//a[contains(text(),'Login')] | | //div[@class='navbar-header']//a[contains(@class,'navbar-link') and contains(@href,'sign\_in')] | | //div[@class='navbar-header']//a[starts-with(@class,'fedora')] | | //a[@href='/sign\_in'] | | //a[@href='/sign\_in']//parent::li//preceding-sibling::li//following-sibling::li[1] | | //div[@class='datepicker-cal-month'][position()=1] | | //div[@class='datepicker-cal-month'][position()=1]//button[@class='datepicker-cal-date' and contains(text(),'22')] | |
| List some Actions on Web Elements? |
| |  |  | | --- | --- | | Click | 1. Button:      1. Radio Button:        1. Checkbox: | | Clear |  | | Sendkeys |  | | getText() |  | | getAttribute(‘attributeName’) |  | | isDisplayed() |  | | isEnabled |  | | isSelected | 1. Radio Button:      1. Checkbox: | |
| List some Actions on drive / browser? |
| |  |  |  | | --- | --- | --- | | **Get** | |  | | **getCurrentUrl** | |  | | **getTitle** | |  | | **getPageSource** | |  | | **findElement** | |  | | **findElements** | |  | | **navigate().to** | |  | | **navigate().back()** | |  | | **navigate().forward()** | |  | | **navigate().refresh()** | |  | | **getWindowHandles()** | |  | | **getWindowHandle** | |  | | **switchTo().window(mywindow)** | |  | | **switchTo().frame(myframe)** | |  | | **switchTo().defaultContent()** | |  | | **switchTo().alert().accept()** | |  | | **switchTo().alert().dismiss()** | |  | | **manage().window().maximize()** | |  | |  | |  | |  |  | | |
| List Some Page Navigation methods? |
| |  |  | | --- | --- | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |
| How to work with dropdown menu with <select> html tag? |
| Few more methods : |
| How can I create my custom method class? |
| Custom class :    Usage : |
| Can you please add method in your custom class to return web element list? |
| Custom Class :    Usage: |
| Please refactor the element list custom method , as it will print element found even if no element is there? |
| Custom Class : |
| Please add one more method in your custom class to, which check if element present with given locator-value? |
| Custom class:    Usage: |
| What are the different types of waits available in Selenium WebDriver With Java Programming Language? |
| There are 2 types of wait  • Implicit Wait  • Explicit Wait |
| What is Implicit Wait? |
| * If elements are not immediately available, an implicit wait tells Web Driver to poll the DOM for a certain amount of time before throwing an exception that it cannot find the element on the page. * The default setting is 0. Once set, the implicit wait is set for the duration of the Web Driver object. |
| Disadvantage of Implicit Wait ? |
| Disadvantages:  (i) In any case, it blindly wait for given seconds.  (ii) Once set, the implicit wait is set for the life of the WebDriver object instance. |
| What is Explicit Wait? |
| * You can simply put a separate time on the required element only until some condition achieved. * WebDriverWait by default calls the ExpectedCondition every 500 milliseconds until it returns successfully. |
| What are some of the expected conditions? |
| These are the available expected conditions:  • elementToBeClickable  • presenceOfElementLocated  • titleContains |
| If you use Implicit Wait of 10 seconds and explicit wait of 20 seconds, how much time Selenium WebDriver will wait before timing out? |
| WARNING:  (i) Do not mix implicit and explicit waits. Doing so can cause unpredictable wait times. For example setting an implicit wait of 10 seconds and an explicit wait of 15 seconds, could cause a timeout to occur after 20 seconds.  (ii) Never ever use both implicit wait and explicit wait on same driver. Its not good practice. Implicit wait is implemented at driver level where as Explicit wait is implemented at language binding level (e.g. Java, c# etc.)  <http://www.seleniumhq.org/docs/04_webdriver_advanced.jsp> |
| Create Custom method class for Explicit wait?  Create Custom method class for Expected conditions? |
|  |
| What is Fluent Wait Command? |
| * Each FluentWait instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. * Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting.      * However these are now available in Explicit wait too |
| What is PageLoadTimeout Command? |
| Sets the amount of time to wait for a page load to complete before throwing an error. If the timeout is negative, page loads can be indefinite. |
| What is SetScriptTimeout Command ? |
| Sets the amount of time to wait for an asynchronous script to finish execution before throwing an error. If the timeout is negative, then the script will be allowed to run indefinitely. |
| How to select calendar items? |
| 1. Using Send keys if possible 2. By Selecting date web element |
| How to select calendar date from element collection? |
|  |
| How to select auto complete element? |
|  |
| What is Java Script Executor ? |
| * In case, Selenium webdriver does not work for any element etc , then you can use JavaScriptExecutor to perform an desired operation on a web element. * Selenium support javaScriptExecutor. There is no need for an extra plugin or add-on. You just need to import (org.openqa.selenium.JavascriptExecutor) in the script as to use JavaScriptExecutor . * JavaScriptExecutor is an Interface that helps to execute JavaScript through Selenium Webdriver. * JavaScriptExecutor provides two methods "executescript" & "executeAsyncScript" to run javascript on the selected window or current page.   Execute JavaScript based code using Selenium Webdriver   1. executeAsyncScript   With Asynchronous script, your page renders more quickly. Instead of forcing users to wait for a script to download before the page renders. This function will execute an asynchronous piece of JavaScript in the context of the currently selected frame or window in Selenium.   1. executeScript   This method executes JavaScript in the context of the currently selected frame or window in Selenium. The script used in this method runs in the body of an anonymous function (a function without a name). We can also pass complicated arguments to it.  The script can return values. Data types returned are  Boolean  Long  String  List  WebElement.  The basic syntax for JavascriptExecutor is given below:  Syntax:  JavascriptExecutor js = (JavascriptExecutor) driver;  js.executeScript(Script,Arguments);  Script – This is the JavaScript that needs to execute.  Arguments – It is the arguments to the script. It's optional.   |  | | --- | | **To type Text in Selenium WebDriver without using sendKeys() method:**  js.executeScript("document.getElementById('some id').value='someValue';");  js.executeScript("document.getElementById('Email').value='SoftwareTestingMaterial.com';"); | | **To click a Button in Selenium WebDriver using JavaScript**  js.executeScript("document.getElementById('enter your element id').click();");  //or  js.executeScript("arguments[0].click();", loginButton); | | **To handle Checkbox**  js.executeScript("document.getElementById('enter element id').checked=false;"); | | **To generate Alert Pop window in selenium**  js.executeScript("alert('Welcome To SoftwareTestingMaterial');"); | | **To refresh browser window using Javascript**  js.executeScript("history.go(0)"); | | **To get innertext of the entire webpage in Selenium**  String sText =  js.executeScript("return document.documentElement.innerText;").toString();  System.out.println(sText); | | **To get the Title of our webpage**  String sText =  js.executeScript("return document.title;").toString();  System.out.println(sText); | | **To get the domain**  String sText =  js.executeScript("return document.domain;").toString();  System.out.println(sText); | | **To get the URL of a webpage**  String sText =  js.executeScript("return document.URL;").toString();  System.out.println(sText); | | **To perform Scroll on application using Selenium**  js.executeScript("window.scrollBy(0,500)");  // for scrolling till the bottom of the page we can use the code like  //js.executeScript("window.scrollBy(0,document.body.scrollHeight)"); | | **To click on a SubMenu which is only visible on mouse hover on Menu**  js.executeScript("$('ul.menus.menu-secondary.sf-js-enabled.sub-menu li').hover()"); | | **To navigate to different page using Javascript**  js.executeScript("window.location = ‘https://www.softwaretestingmaterial.com”); | |
| What is the javascript executor for open a url? |
| js = (JavascriptExecutor) driver;  js.executeScript("window.location = 'https://letskodeit.teachable.com/pages/practice';");  Note:  driver.get() method waits for the page to load completely before going to the next statement , js.executeScript() does not wait for the page to load completely |
| What is the javascript executor to get an element? |
| WebElement textBox = (WebElement) js.executeScript("return document.getElementById('name');"); |
| What is the javascript executor to get height and width of window? |
| long height = (Long) js.executeScript("return window.innerHeight;");  long width = (Long) js.executeScript("return window.innerWidth;"); |
| How can you scroll to an element? |
| WebElement element = driver.findElement(By.id("mousehover"));  js.executeScript("arguments[0].scrollIntoView(true);", element); |
| What is the javascript executor to click ? |
|  |
| What is the javascript executor to sendkeys ? |
|  |
| How to hover mouse ? |
| Actions action = new Actions(driver);  action.moveToElement(mainElement).perform(); |
| How to drag and drop an element |
| Actions action = new Actions(driver);  // Drag and drop  action.dragAndDrop(fromElement, toElement).build().perform();  // Click and hold, move to element, release, build and perform  action.clickAndHold(fromElement).moveToElement(toElement).release().build().perform(); |
| How to use slider ? |
| Actions action = new Actions(driver);  action.dragAndDropBy(element, 100, 0); |
| How to perform keyboard operations? |
|  |
| How to send multiple keys operations? |
|  |
| Perform Keyboard options using action class ? |
|  |
| File upload using Robot class ? |
| String filePath = "C:\\Users\\Vish\\Desktop\\testfile.txt";  StringSelection stringSelection= new StringSelection(filePath);  //Copy to clipboard  Toolkit.*getDefaultToolkit*().getSystemClipboard().setContents(stringSelection, null);  Robot robot = new Robot(); |
| File upload using copy paste to AutoIT ? |
| 1. Download the full installation version of AutoIT      1. After installation, use AutoIt Window Info – Finder tool to find the control Ids        1. Open SciTE Editor to write script and save it as .au3 file. 2. Right click on the au3 file and compile it to .exe file.      1. Now invoke .exe file from your test |
|  |
|  |
|  |
|  |
|  |