**Ports**

* Default port used by Jenkins – **8080**
* Default port for Selenium hub – **4444**
* Default port used by Selenium for communication – **4444**
* Default port for Selenium Node for communication – **5555**
* HTTP server of the IE driver will listen for commands from language bindings. Defaults to **5555.**
* Gecko Driver - Port to use for the WebDriver server. Defaults to **4444**

**Remote WebDriver:**

Remote Web Driver compose of **2 pieces:**

* A Client
* A Server
* **Client mode:** where the language bindings connect to the remote instance. This is the way that the FirefoxDriver, OperaDriver and the RemoteWebDriver client normally work.
* **Server mode:** where the language bindings are responsible for setting up the server, which the driver running in the browser can connect to. The ChromeDriver works in this way.

**Advantage:**

* Allows test to be run with browsers not avaiable on the current OS
* Separates where the tests are running from where the browser is

**Disadvantage**

* Requires an external servlet container to be running

**DOM – Document Object Model**

* It is HTML doc and can be accessed using JavaScript

**Default timeout for Selenium Grid: 300 Secs**

* 300 secs is default timeout for selenium grid before the hub automatically releases a node that hasn’t received any request for more than specified number
* SafariDriver implemented using – **Extensions**
* **Global Extension** of SafariDriver is responsible for communicating with the WebDriver Client
* **userExtension** can be given to selenium Server during startup for loading Javascript file
* **Firebug** in Selenium is used for **Inspecting Elements**
* **Page Inspector** is used to inspect elements and is also a built-in feature of Firefox browser after version 47.
* **AJAX – Asynchronous JavaScript and XML**
* **JSON - JavaScript Object Notation**
* Peer to Peer in HTML5 -technology -**WebRTC**
* Hierarchical value stored in HTML5 **– IndexedDB**
* Elements are used to draw charts, graphics and data driver program in HTML5 - **Canvas**
* Under the banner of **HTML5**, modern web standards such as **CSS3, SVG, XHR2, WebSockets, IndexedDB, and AppCache** are pushing the boundaries for what a browser can achieve using web standards.
* **DOM** **Query i**s used to locate element in JavaScript
  + **DOM** query is used for Evaluating / finding element on the page
  + **Xpath** query is use to identify nodes in XML document
  + **CSS Selector** is a selection of code that is use to define style property of web page
* **By DOM Query** process uses JavasScript to find an element
* **In Selenium 2 ways for storing value into Variable**
  + store(expression, variable name) – for storing value
  + storedVars[‘variable name’] - for storing one variable value to another variable
  + **Format**
    - **Stored[‘Variable name’]**
    - **${variable name}**
* Selenium cannot access elements outside of the web application under test.
* All variables created in test cases are stored in a Java script **Associative Array .** Associative Array has **string index** rather than sequential numeric index.
* In Javascript, **executeScript()** is used to move to a web Element
* **WebDriverEventListener** is provided in WebDriver to track events those take place in WebDriver during **script execution.**
* **WebDriver** class does EventFiringWebDriver, acts as an wrapper on
* **-D** command line flag is used by Firebox driver to change system property
* **HtmlUnitWebDriver** is used for **Headless browser testing**
* **fireEvent()**command is a “OnEvent Handler”.
* Why is the actions class used for in WebDriver? **It controls the actions of mouse**
* Action is a class used to **perform mouse actions** in Selenium WebDriver
* **verifyTable** command is used to compare the contents of a table with expected values
* **Anonymous** is the default profile created by Firefox driver
* **HTMLUnit** Support feature is additional in Selenium 2

**Jenkings**

* **WAR file -** The **Web application ARchive (WAR)** file version of Jenkins can be installed on any operating system or platform that supports Java
* Which Jenkins job type, is usually selected for integration with Selenium **–** **Free Style Project**
* Jenkins accepts **5 parameters** for periodic execution
* Minimum **Java 8 version** needed for **Jenkins installations**
* What is the Java file format of Jenkins installer usually used? WAR file.
* **Jenkins benefits**
  + Free and open source tool
  + Widely used with good documentation
  + Seamless integration with many tools and technology

**TestNG & Thread**

* TestNG - **TestNextGeneration**
* **TestNG framework allows us to generate test reports in both HTML and XML formats.**
* In TestNG, **each instance or suites to be run in a separate thread**
* **thread-count** attribute specifies **how many threads** should be allocated when parallelizing test runs in Selenium with TestNG
* **Same thread is allocated to methods in the same class**, if using classes as value for parallel attribute, while parallelizing test runs in selenium with TestNG
* **Separate Thread is allocated to each class** , if using classes as value for parallel attribute , while parallelizing test runs in Selenium with TestNG
* **Separate thread is allocated to dependent methods**, if using methods as value for parallel attribute , while parallelizing test runs in Selenium with TestNG
* **Separate thread is allocated to all of the test methods**, if using methods as value for parallel attribute , while parallelizing test runs in Selenium with TestNG
* parallel=”methods”: TestNG will run all your test methods in separate threads. Dependent methods will also run in separate threads but they will respect the order that you specified.
* parallel=”tests”: TestNG will run all the methods in the same <test> tag in the same thread, but each <test> tag will be in a separate thread. This allows you to group all your classes that are not thread safe in the same <test> and guarantee they will all run in the same thread while taking advantage of TestNG using as many threads as possible to run your tests.
* parallel=”classes”: **TestNG will run all the methods in the same class in the same thread**, but each class will be run in a separate thread.
* parallel=”instances”: TestNG will run all the methods in the same instance in the same thread, but two methods on two different instances will be running in different threads.
* Parallel execution of Selenium test cases is possible in TestNG
* TestNG has a more elegant way of handling parameterized tests with the **data-provider concept.**
* **Features of TestNG**
  + Support for parameterization
  + Support for Data Driven Testing using Data providers
  + Enables user to set execution priorities for the test methods
  + Facilitates user with effective means of Report Generation using ReportNG
  + Supports threat safe environment when executing multiple threads
  + Support for annotations
* The Time-out test in TestNG - It's the time duration to wait for a test to finish its execution
  + TestNG doesn't need to extend any class whereas JUnit docs
  + The @DataProvider annotation accepts single string attribute and yields back an array of objects.
  + TestNG permits to define the dependent test cases. Each test case is independent of other test cases
* Common usage of wrapping Selenium methods is to check for presence of an element on page before carrying out some operation. This is sometimes called a ‘**safe operation’**. For instance, the following method could be used to implement a safe operation that depends on an expected element being present.
* Jenkins supported OS – Linux, Mac OS and Windows
* Mavan supported OS – Linux, Mac OS and Windows
* Selenium WebDriver Supported OS - Linux, Mac OS, Windows and Solaris.
* AutoIT Supported OS – Windows
* Minimum Firefox version needed for geckodriver? **Version 48**
* **Selenium WebDriver Supported browsers –** Firefox, IE, Chrome, Safari, Opera 11.5 & above, Android, IOS, Htmlunit 2.9 & above
* **JAVA\_HOME** environment is essential in integrating Selenium and Mavan
* Minimum version of JDK needed to install latest **Mavan installation** is – **JDK Version 1.7**
* Default folder name for Mavan is **.m2**
* Mavan is a Java based tool and it needs JDK 7 or above for installation
* **Surefire Plugin -** The Surefire Plugin is used during the test phase of the build lifecycle to execute the unit tests of an application
  + Generate reports in the form of text file or xml file
* **FireEvent** is used to customizing events in Selenium
* **webdriver.ie.driver** is system property used by the InternetExplorerDriver , to specify the location of the ID Driver binary.
* Howmany values does parallel attribute of suite tag can accept, when parallelizing test runs in Selenium with TestNG? - **4 Values**
* **ANT** is a build tool

**Selenium History**

* **Jason Huggins** was developed Selenium in 2004
* **Simon Stewart** was developed WebDriver in 2006
* Selenium and WebDriver were merged in **2008**
* Selenium 3 official releases made on 2016
* Selenium 1.0 supports **Mobile Web Application and Computer Web Applications**
* Selenium 1 aka **Selenium RC**
* Selenium 2 aka **Selenium WebDriver**
* **Selenium 2 is browser specific implementation.**
* Minimum version of Java required to run **Selenium 3** is **Java 8+**
* Earliest version of Selenium was developed in **JavaScript**

**Selenium RC**

* **Selenium RC Server** is an packaged **Java Jar file**
* **Browserbot** is the name of the Javascript object which wraps the DOM in Selenium RC.
* Selenium RC is used to **To run tests in different browsers (except HtmlUnit) on different OS**
* **Selenium RC** is a Javascript based automation engine or sandbox,as it needs to comply with Same original policy
* **Host** is the parameter name which specifies the IP address of the computer where the server is located,when creating browser instance in Selenium RC
  + If same machine as where client is running then localhost is passed
  + Browser stores browser details

**Selenium IDE**

* **Selenium IDE is FireBox Add-ins**
* **Selenium IDE** is an **XPI file which is an Firefox add-in** file
* **Selenium IDE** not supported **DB/2 and Unix OS**
* **Selenium IDE does not support any programming.**
* **Selenium IDE will NOT work on Firefox version 55 onwards.**
* Selenium IDE supporting recording, batch testing and test execution
* **Open** command in Selenium IDE is used to open a page using the URL
* **Selenium IDE Advantages:**
  + Open Source and multiple execution at a time
  + Supports all browsers and OS
  + Intelligent field selection
  + Support user extension
  + Auto Complete commands use ID, Name, Xpath and CSS Selector
* **Target** column in Selenium IDE has a find button in input
* **Command** column provides autocompletion by drop down list of values during input in Selenium IDE
* A Selenium IDE test case has 3 columns Command, Target and value. What data is stored in the Target Column? **Element or location where the command is executed**
* **HTML** is the default file format does the Selenium IDE save a test case
* Test script in source viee of Selenium IDE is – **HTML format**
* In case of Selenium IDE, the Source view shows your script in - XML format (doubt)

**Selenium Grid**

* Default timeout value for Selenium hub release a node automatically **– 300 seconds**
* In Selenium grid – maximum number of browsers that can run in parallel by default – **5 browsers maxSession 5 (5 is default)**
* Selenium grid config file format **– JSON**
* **Selenium hub** is the component in Selenium grid which routes the selenese request from the test to the appropriate Selenium Remote Control.
* **There is no limit** of usuage of WebBrowser are available for concurrent usuage after starting the node in Selenium grid
* **Apache Ant,** an Java build tool is needed for Selenium grid.
* **HTTP** protocol is used for communicate between hub and node in Selenium grid.
* browserName={android, chrome, firefox, htmlunit, internet explorer, iphone, opera}
* By default, starting the node allows for concurrent use of 11 browsers… : 5 Firefox, 5 Chrome, 1 Internet Explorer. **The maximum number of concurrent tests is set to 5 by default.**
* What is the default value for, number of Firefox web browsers, available for concurrent usage after the starting node in selenium grid? **5 browsers**
* A grid consists of a single hub, and one or more nodes. Both are started using the **selenium-server.jar** executable.
* In 2.0 Selenium-Grid was merged with the Selenium-RC server. Now, you only need to download a single .jar file to get the remote Selenium-RC-Server and Selenium-Grid all in one package
* The Remote Control instances that have been registered with the grid show up in <http://localhost:4444/grid/console>.
* Selenium Grid does not expose any APIs
* Main advantage of selenium grid is Executing parallel tests

**Selenium 3**

* In Selenium 3, all the major browser vendors (Apple, Google, Microsoft, and Mozilla) ship a WebDriver compatible to their browser.
* Safari driver will get native support from Apple on MacOS (Sierra or later).
* JRE 8.0 is the minimum requirement to run Selenium 3.0.
* No out of the box support for Firefox Webdriver after 47.0.1. Use Geckodriver for newer versions
* Released on 2016
* Firefox 47.0 not supported at all
* Minimum Java 8+ is required
* Apple has come up with its own SafariDriver to let you run your tests in Safari on Mac

**WebDriver**

* + WebDriver is a compact Object Oriented API
  + WebDriver drives the browser much more effectively
  + WebDriver Overcomes the limitation of single-host origion policy
* WebDriver is an Interface
* **WebDriver over Selenium RC**
  + WebDriver does not need the Selenium RC server to be running
  + WebDriver has native web browser support and runs faster than RC
  + WebDriver supports headless HTMLUnitDriver that enables faster test execution
  + WebDriver is a compact Object Oriented API
  + WebDriver drives the browser much more effectively
  + WebDriver Overcomes the limitation of single-host origion policy
* **WebDriver** is a set of native APIs which directly send commands to the browser

instead of delegating to a server

* **Recording** is not supported by Selenium WebDriver
* **HTTP** is the main transport machanism used by WebDriver.
* **WebDriver’s** Actions commands are**directly interact with page elements**
* **Htmlsuite** command argument is used to run Selenium html files directly within the selenium server by passing the html file to the server’s command line
* **Perl**  Language bindings are Third party lanuuage bindings ad NOT developed by SeleniumHQ.
* **waitForPageToLoad** command waits for the web page to load
* **waitForElementPresent** command will enable the test case to be passed, after addition of a new element post page load.
* Selenium variables are accessed by prefixing **$** and enclosing braces eg: **${name}**
* **APIs** are the methods to add plug-ins in Selenium.
* **Apache Ant** is a software **build tool** for automating software build process.
* **JUnit 5** = JUnit Platform + JUnit Jupiter + JUnit Vintage
* In Junit **Failure** is **AssertionException (assertion has occurred)**
* In Junit **Error** is **Exception has occurred**
* **When a JUnit test is** "**Failed**", you have gotten an **AssertionException.** Means in your case result was false, where it should have been true.
* **ErrorInResponseException** is thrown when an error has occurred on the server side
* Junit reporting for tests with Selenium RC is done in **XML** format.
* **Assert** f**ails,** the test will be aborted and will **stop execution**
* **Verify fails**, the **test will continue** and **logging the failure**
* **verify is**  usually used to check non-critical things on a web page
* **assert** is used to check critical things on a web page, whose failure should not lead to continuationebent the test
* **Making assertion** is usually avoided in page object pattern
* **Echo()** command gives its output to **log file.**
* **ECHO()** command is used to display stored value of variable in the log file. The command is specially used at the time of debugging process.
* **Exit** command is used to exit from any execution
* **NUnit** is an **.Net test** engine
* **Junit, NetBeans and IntelliJ** are Java related
* **Dynamic testing** test design technique is applied in unit, integration &System testing.
* **Static testing** test design technique is applied in Inspection, Review & walkthrough
* **//** shortening of Xpath for accessing an element.
* **JXL/POI** are the external jars used to import or include any excel sheet.
* TestNG and Junit are framework.
* **Glob** is the default matching mechanism for string matching pattern in Selenium
* The below are the 2 main classes of **WebDriverAPI ,** if testing under **Ruby**
  + **Selenium::WebDriver::Driver**
  + **Selenium::WebDriver::Element.**
* Perl bindings are provided by a third party. Perl modules that are needed to make it work are
  + **Selenium::Remote::Driver**
  + **Test::More**
* **Selenium::Remote::Driver** is the name of the **perl module** is **used to connect Selenium Server (Hub/Node) remotely** and send command to the WebDriver API
* **Test::More –**framework for writing test scripts. It will generate TAP test results which can be integrated into Jenkins for CI (continues integration) or Nagios for monitoring test results in Perl
* Howdoes the webdriver interact withs browsers for test execution? WebDrievr API
* **Timout** parameter in remote webdriver, controls howlong the client is allowed to be gone before the session is reclaimed.
* **Send() in AutoIT** used to send keyboard keys to the AUT
* **getRuntime** – AutoIT function is used to get the current runtime associate with the selenium process.
* **exec** AutoIT function is used to execute an executable progam of AutoIT script
* **runScripts** adds Specific script tag
* **FlexUISelenium** used to test Flex/Flash applications using Selenium
* There is no limit on number of **Popups** to be managed **by Selenium.**
* Which type of Selenium calls are wrapped in a wrapper method? – **Frequently used Selenium calls**
* **Normal** is the default page load strategy followed by **geckodriver**
* **NuGet** is usually used for installing selenium for C# project
* **Static testing** used in **reviewing, walk through, inspection**,
* **Dynamic Testing** used in **Unit testing, integration testing, system testing**.
* **AJAX Tests** supports dynamically changing user interface elements which can dynamically change without the browser having to reload the page. Such as **animation, RSS feeds, and real-time data updates** among others
* **Map** data structre is used to provide **ChromeOptions** by ChromeDriver in Java project
* How does UI map properties file contains UI element? **Key in Key-value pair**
* How does UI map properties file contains the **locator** of UI element? **Value in Key-value pair**
* A **UI map** is a mechanism that stores all the **locators** for a test suite in one place for easy modification when identifiers or paths to UI elements change in the AUT
* UI Map provides below functions for UI element of an application
  + Defining UI element
  + Stroing UI element
  + Serving UI element
* **Page Object** Model is a design pattern to create Object Repository for web UI elements in Page object pattern
* Page object pattern is based on **Abstraction** OOPS concept
* Which object prototype in Selenium, is used for extending Selenium? PageBot Object Prototype
* According to which WebDriver implementation claims to be the fastest? HTMLUnitDriver claims to be the fastest implementation of WebDriver is because the HTMLUnitDriver does not execute tests in the browser.
* **Internet Explorer web browser** is having idiosyncracies.related to fileUploadDialof Timeout
* Default fileUploadDialog timeout in IEDriver is **1000ms**
* **Windsows** OS supports Selenium installation and application on IE web browser.

**Wait**

* **Wait** command is used to pause or wait the execution process for specific time
* **WaitForProperty** command is used to pause the execution process until any specific value
* **Implicit Wait** objects keep alive until the WebDriver object dies
* **WebDriverWait** by default calls the ExpectedCondition every **500 milliseconds** until it returns successfully
* **waitFor** command is used for handling AJAX controls by selenium
  + waitFor command waits dynamically and checking for the desired condition every second and continue next command in the script as soons as the condition is met.
* **SetScriptTimeout :** It 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.
  + Example –  driver.manage().timeouts().setScriptTimeout(100,SECONDS);
* **WebDriver wait by default** calls the ExpectedCodition every **500 milliseconds** until it returns successfully.
* Default polling interval is 500 ms
* **Fluent wait initialize required**
  + The maximum amount of time to wait for a condition.
  + The frequency to check the success or failure of a specified condition.
  + Exception classes to ignore while waiting eg : NoSuchElementExceptions

**Logs**

* **Debug** WebDriver log level is used to show all message for debugging like information about the state of the driver
* **Warning** webdriver log is used to show messages only about things that may be wrong but was handled
* **Severe** log type is used to show message only about things that went wrong like an unknown command
* **Fatal** is default log level at which logging message are output by the InternetExplorerDriver
* **Debug** is given to selenium server for diagnostics.
* **Web Browser** log type collects Javascript console logs from the browser
* **Client** log type collects logs for Java bindings under Selenium
* **Server** log type collects logs from within the selenium server.
* **Driver** log type collects logs for FirefoxDriver internals under Selenium
* **Info** log type listed in log pane of Selenium IDE , shows which command Selenium IDE is currently executing
* **Reference** tab which gives feedback and other useful information when executing tests
* **Performance** log type collects logs for resource loading.
* **Performance logtype** in Selenium, logs relating to the performance characteristics of the page under test (i.e resource load timings)
* **Off** webDriver log level is used to turn off logging**.**
* **Stdout** is the default path and file name of the log file used by the InternetExplorer Driver
* what is the default value for path to file where selenium server should write log messages to? **Stdout**
* what is the default value for path to directory used to extract supporting files used by the serverin InternetExplorer Driver? **temp**
* **LOG Types:**
  + **Browser -** Javascript console logs from the browser
  + **Client -** Logs from the client side implementation of the WebDriver protocol (e.g. the Java bindings)
  + **Driver -** Logs from the internals of the driver (e.g. FirefoxDriver internals)
  + **Performance -** Logs relating to the performance characteristics of the page under test (e.g. resource load timings)
* **LOG Levels**
* **OFF**: Turns off logging
* **SEVERE**: Messages about things that went wrong. For instance, an unknown command.
* **WARNING**: Messages about things that may be wrong but was handled. For instance, a handled exception.
* **INFO**: Messages of an informative nature. For instance, information about received commands.
* **DEBUG**: Messages for debugging. For instance, information about the state of the driver.
* **ALL**: All log messages. A way to collect all information regardless of which log levels that are supported.
* **Log entry contins**
* Timestamp , log level and message
* The below are needed for development and testing for **Python project with Selenium**:
  + PIP
  + Setuptools
  + Python driver
* What are the features available in **Selenium IDE to dubug** an automated test cases?
  + Step
  + Pause / Resume
  + Toggle Breakpoint

**User Extension:**

* + User Extensions which are in the form of JavaScript files.
  + 3 main pillars
    - **Action:** What operation you are performing on UI Screen
    - **Assessors/Assertion:** What verification you do on data you get from UI
    - **Locator Strategy:** How can we find the element in UI.
  + **Locator** is used to extend selenium and refers to finding the element in UI
  + In UI map machanism that stores all the **locators** for that test suite in one place for easy modification when identifiers or paths to UI elements change in the AUT
  + **What aspact of an element are stored in a UI map?**
    - Name of the element
    - Locator type of element
    - Value of the locator
  + **Action**
    - All methods on the Selenium prototype beginning with **“do”** are added as actions.   It can accept **two parameters, locator, text**
      * Eg : Selenium.prototype.doTextUpperCase = function(locator, text)
  + **Assessors/Assertion :** prototype will be prefixed by “**get” or “is”.** .It can accept **two parameters**, one for **target** and other for **value field in test case**.
    - For each Assessor there will be corresponding verification functions prefixed by “verify”, “assert” and the wait function prefix by “waitFor”
      * **Eg :** getValueFromCompoundTable , isValueFromCompoundTable
    - For each accessor there is an **assertFoo, verifyFoo and waitForFoo** registered.
    - An assert method can take up to 2 parameters, which will be passed the second and third column values in the test.
  + **Locator strategy :** we need to extend **PageBot prototype** with a function with prefix “l**ocateElementBy**”. It will take **two parameters**, first will be the **locator** string and second will be the **documen**t where it needs to be search,

|  |  |  |
| --- | --- | --- |
| **User Extension** | **First parameter** | **Second Parameter** |
| **Action** | Locator | Text |
| **Assessors/Assertion** | Target | Value field in test case |
| **Locator strategy** | Locator | Document where it needs to search |

**Time**

* The command that is being used to extend the time limit of WAITFOR command is **setTimeout**
* Default time for WAITFOR command is **30 secs.**
* WaitFor command is used to pause or wait the execution for specified time. Default time 30 secs. But we can change the time period by using setTimeout command

**Axis**

* **Ancestor** – select all the element of the parent node, grantparent node and all other nodes of the system
* **Preceding** - it contains all the nodes those are already occur in context or system
* **Parent** -it only contains the parent node of the context or system
* **Following** : Select all elements that follow the enclosing tab of the current elements

**Alert**

* **storeAlert** command is used to retrieve the alert message and thereby stores it in a variable beging specified.
* **Web based Alerts managed by WebDriver**
  + accept()
  + getText()
  + dismiss()
  + sendkeys(String stringtosend)

**Page object Model**

* Advantage
  + Easy to maintain
  + Clean and understandable code
  + Better test Scripts

**Mobile WebApp Testing**

* WebDriver has **two main components**: the **server** and the **tests** themselves. The server is an application that runs on the **phone, tablet, emulator, or simulator and listens for incoming requests**. It runs the tests against a WebView (the rendering component of mobile Android and iOS) configured like the browsers. Your tests run on the client side, and can be written in any languages supported by WebDriver, including Java and Python. The WebDriver tests communicate with the server by sending **RESTful JSON requests over HTTP**.

Relative XPath with Starting Text

* + //button[starts-with(@id, ’Vskills-’)]

Relative XPath with Following or Preceding Node

* //button [contains(@class, ‘Vskills-Class’)] /**following::** input[contains(@id,’Vskills-’)]
* //input [contains(@id,’Vskills-’)] /**preceding::** button[contains(@class, ‘Vskills-Class’)]

Relative XPath with Multiple Attribute

* //button[contains(@id,’Vskills-’)] [contains(@class, ‘Vskills-Class-text’)]

What will be the CSS Selector equivalent for a sub-child in xpath expression //div//a?

**Direct child**

**XPath: //div/a  
  
CSS: div > a**

A direct child in XPATH is defined by the use of a “/“, while on CSS, it’s defined using “>”

**Child or subchild**

**XPath: //div//a   
  
CSS: div a**

If an element could be inside another or one it’s childs, it’s defined in XPATH using “//” and in CSS just by a whitespace

**Choosing a specific match**

<ul id = "recordlist">

<li>Cat</li>

<li>Dog</li>

<li>Car</li>

<li>Goat</li>

</ul>

If we want to select the fourth li element (Goat) in this list,

CSS: **#recordlist li:nth-of-type(4)**

**Sub-string matches**

CSS in Selenium has an interesting feature of allowing partial string matches using ^=, $=, or \*=. I’ll define them, then show an example of each:

^= Match a prefix

**CSS: a[id^='id\_prefix\_']**

A link with an “id” that starts with the text “id\_prefix\_”

$= Match a suffix

**CSS: a[id$='\_id\_sufix']**

A link with an “id” that ends with the text “\_id\_sufix”

\*= Match a substring

**CSS: a[id\*='id\_pattern']**

A link with an “id” that contains the text “id\_pattern”

### **Matching by inner text**

**CSS: a:contains('Log Out')**

Selenium programming language and testing frameworks, are

* C# - NUnit
* Haskell
* Java - JUnit, TestNG
* JavaScript - WebdriverJS, WebdriverIO, NightwatchJS, NemoJS
* Objective-C
* Perl
* PHP - Behat + Mink
* Python - unittest, pyunit, py.test, robot framework
* R
* Ruby - RSpec, Test::Unit

**Read data from Excel:**

* **Jxl.jar** is an open source Java API which supports To read data from Excel using selenium webdriver
* Apache POI is an API, which is freeware and written in Java
* JXL and Apache POI APIs have the same end functionality
* **JXL/POI** are the external jars used to import or include any excel sheet.

// Specify the path of file

**File src=new File(“filepath/excelsheetname.xlsx”);**

// load file

**FileInputStream fis=new FileInputStream(src);**

// Load workbook

**XSSFWorkbook wb=new** **XSSFWorkbook(fis);**

// Load sheet- Here we are loading first sheet only

**XSSFSheet sh1= wb.getSheetAt(0);**

// getRow() specify which row we want to read.

// and getCell() specify which column to read.

// getStringCellValue() specify that we are reading String data.

//getRows() – Specify number of rows

* What provides a handle to the inidividual cells, or lines of cells (grouped by Row and Column) in JXL? **Sheet**
* Apache POI, is capable enough to read and write both **XLS and XLSX** file format of Excel.
  + To read **XLS files, an HSSF implementation** is provided by POI library.
  + To read **XLSX, XSSF implementation** of POI library will be the choice
* **Open a excel for read**
  + **File myxl=new File()**
  + **FileInputStream fis=new FileInputStream(myxl);**

**Cookies**

* **HTTP Cookie** is also called as a **web cookie, a browser cookie** or an **Internet cookie**.
* **HTTP Cookies stores**
  + user’s browsing history
  + shopping cart information for an online store,
  + login user ids and passwords
  + and other related information
* **getcookie** method is used to access Cookies.

driver.manage().getCookies(); // Return The List of all Cookies

driver.manage().getCookieNamed(arg0); //Return specific cookie according to name

driver.manage().addCookie(arg0); //Create and add the cookie

driver.manage().deleteCookie(arg0); // Delete specific cookie

driver.manage().deleteCookieNamed(arg0); // Delete specific cookie according Name

driver.manage().deleteAllCookies(); // Delete all cookies

* **getCookies()** method extracts a cookie and parse it.
* **Getcookies** method extract a cookie and parses it.
  + Get all the cookies for the current domain. This is the equivalent of calling “document.cookie” and parsing the result.
* **getCookieNamed(arg0);**
  + It will return the cookie value for the name specified, or null if no cookie found with the given name
* **deleteCookieNamed(arg0);**
  + Delete the named cookie from the current domain. This is equivalent to setting the named cookie’s expiry date to sometime in the past.
  + Void deleteCookieNames(String Name)
* We can delete cookies in 3 ways.
  + driver.manage().deleteCookieNamed("CookieName");
  + driver.manage().deleteCookie(CookieId); or driver.manage().deleteCookie(loadedCookie);
  + driver.manage().deleteAllCookies();

**Pattern Match in Regular Expression:**

**.** Any Single character

Asterisk \* - 0 or more character (nothing or anything)

+ - 1 or more characters

**Maximum Parameter passed in the function:**

* clickAndHold() – 1
* isDisplayed() – 0
* dragAndDrop() – 2
* dragAndDropBy() - 3

**Number of Argument Accepted by the function:**

* afterNavigateRefresh(WebDriver driver) - 1
* beforeNavigateRefresh(WebDriver driver) – 1
* afterNavigateBack(WebDriver driver) – 1
* beforeNavigateBack(WebDriver driver) – 1
* afterNavigateForward(WebDriver arg0) – 1
* beforeNavigateForward(WebDriver arg0) – 1
* afterChangeValueOf(WebElement arg0, WebDriver arg1) – 2
* beforeChangeValueOf(WebElement arg0, WebDriver arg1) – 2
* afterClickOn(WebElement arg0, WebDriver arg1) – 2
* beforeClickOn(WebElement arg0, WebDriver arg1) – 2
* afterNavigateTo(String arg0, WebDriver arg1) – 2
* beforeNavigateTo(String arg0, WebDriver arg1) **: void** – 2
* afterScript(String arg0, WebDriver arg1) - 2
* beforeScript(String arg0, WebDriver arg1) – 2
* onException(Throwable arg0, WebDriver arg1) – 2
* afterClickOn(WebElement element,  WebDriver driver)-2
* beforeClickOn(WebElement element,   WebDriver driver) -2
* afterScript(java.lang.String script, WebDriver driver) - 2
* beforeScript(java.lang.String script,  WebDriver driver) – 2
* afterFindBy(By arg0, WebElement arg1, WebDriver arg2) – 3
* beforeFindBy(By arg0, WebElement arg1, WebDriver arg2) – 3

# [Selenium Grid: MaxSessions vs MaxInstances](https://stackoverflow.com/questions/13723349/selenium-grid-maxsessions-vs-maxinstances)

# MaxInstances  how many instances of same version of browser can run over the Remote System.

For example, i have a FF12,IE and i declared the command as follows

-browser browserName=firefox,version=12,maxInstances=5,platform=LINUX

-browser browserName=InternetExplorer,version=9.0,maxInstances=5,platform=LINUX

So i can run 5 instances of Firefox 12 and as well as 5 instances of IE9 at the same time in remote machine. So total user can run 10 instances of different browsers (FF12 & IE9) in parallel.

# MaxSession  how many browsers (Any Browser and any version) can run in parallel at a time in the remote system. So this overrides the Max Instances settings and can restrict the number of browser instances that can run in parallel.

For above example, when maxSession=1 forces that you never have more than 1 browser running.

With maxSession=2 you can have 2 Firefox tests at the same time, or 1 Internet Explorer and 1 Firefox test).

**How can we capture Screenshot in Selenium WebDriver?**

selenium offers an interface referred as TakesScreenshot that has a method referred as 'getScreenShotAs' that can be used to take a screenshot of the application under test. Now in Selenium 3, there is a possibility of certain issues while capturing Screenshots. In order to overcome this situation we may use aShot utility.

https://www.techbeamers.com/top-30-selenium-webdriver-interview-questions/

* Function libraries concept **can be** implemented in Data Driven Framework.
* Data Driven Framework supports parallel execution of scripts.
* **Keyword Drivern Framework** requires development of data tables and is independent of the test automation tool
* **JXL/POI** are the external jars which will be required to develop Keyword Drivern Framework
* **Checked type** of exception is handled during **compile time**
* What is the data type of first argument of enable function for logging under Java in Selenium WebDriver – **String**
* **Selenium.setSpeed()** - Runs each command after setSpeed delay by the number of milliseconds specified in setSpeed().
* **Thread.sleep()** - Waits for only once at the command given at sleep.

**getWindowHandle() vs getWindowHandles()**

|  |  |
| --- | --- |
| **getWindowHandle()** | **getWindowHandles()** |
| will provide a unique identifier(handle) of the current browser window which is being controlled by the WebDriver | provide a set(collection) of all existing window identifiers(handles) present at a given time |
| return type is string | return type is Set<string>. |
| used to handle single window i.e. main window | used to handle multiple windows |
| gets the address of the current browser | returns handles of all the open browsers |

* **Data Source is not a valid component of an UI automation framework**
* **Component of an UI automation framework:**
  + Object Repository
  + Reporting and logging
  + XML Parser
  + Configuration File
* **keyword driven framework** - It is best suited for **functional testing** and is a **functional automation testin**g framework
* The **default status of JavaScript in HTMLUnitDriver** – Disabled
* To enable JavaScript in HTMLUnitDriver during driver initialization –
  + HtmlUnitDriver driver = new HtmlUnitDriver(true);
* **extern** keyword not associate with Java

**There are three kinds of exceptions:**

1. Checked Exception

2. Unchecked Exception

3. Error

#1**) Checked Exception**: Checked exception is handled **during compile time** and it gives the compilation error if it is not caught and handled during compile time.

Example: FileNotFoundException, IOException etc.

#2) **Unchecked Exception:** In case of the unchecked exception, a **compiler does not mandate to handle**. The compiler ignores during compile time.

Example: ArrayIndexoutOfBoundException

#3) **Error:** When a scenario is fatal and the program cannot recover then JVM throws an error. Errors cannot be handled by the try-catch block. Even if the user tries to handle the error by using Try catch block, it cannot recover from the error.

Example: Assertion error, OutOfMemoryError etc.

there are many Exception classes under WebDriverException, we commonly see the below ones.

· NoSuchElementException

· NoSuchWindowException

· NoSuchFrameException

· NoAlertPresentException

· InvalidSelectorException

· ElementNotVisibleException

· ElementNotSelectableException

· TimeoutException

· NoSuchSessionException

· StaleElementReferenceException

**Page Object Model Advantage**

* POM provides clean separation between test code and page specific code.
* POM solves the problem of duplicate locators for same WebElement.
* POM reduces the maintenance of the test scripts
* POM ensures code re-usability.
* How Different results for the same action are modeled in page object pattern?

**As different method**

* Important factor which are responsible for using Selenium-Server with Selenium WebDrriver
  + Not using java binding (eg: Python,C#,Ruby) and like to use HTMLunit Driver
  + Connect to a remote machine
  + Using Selenium-grid to distribute test to multiple machines or VMs.
* **Web Application cater**
  + B2C or E-commerce
  + B2B or trading web sites
  + Organizationals internal needs
* Which column values in the test table will be taken by first argument of custom action used for extending Selenium? **Second column**
* Which column values in the test table will be taken by second argument of custom action used for extending Selenium? **Third column**
* An action method can take upto 2 parameters. Which will be passed the second and third column values in the test.
* **Test Automation**
  + **Advantage:**
    - Rapid feedback to developers
    - Frequent regression testing
    - Virtually unlimitted iterations of test case executions
    - Displined documentation of test cases
    - Support Agile and extreme development methodology
    - Customized defect reporting
  + **Disadvantage**
    - Profeciency is required to write automation script
    - Debugging the test script is major issue
    - Test maintenance is costly incase of playback mode
* **Changing the User Agent in Java**
  + FirefoxProfile profile = new FirefoxProfile();
  + profile.addAdditionalPreference(“general.useragent.override”, “some UA string”);
  + WebDriver driver = new FirefoxDriver(profile);
* What is needed for development and testing for **Ruby** project with selenium?
  + YARV
  + RubyGems
  + Ruby driver

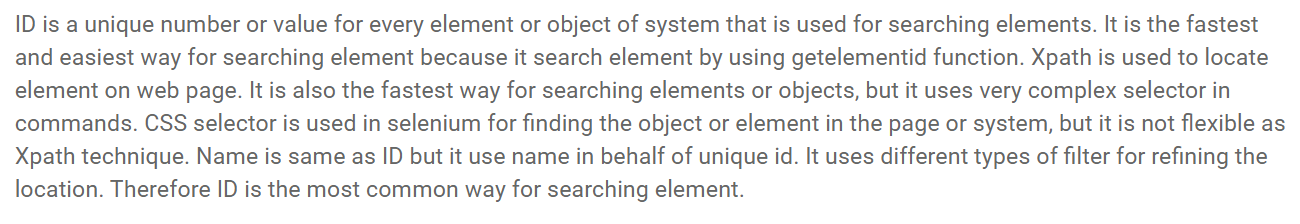
**InternetExplorer WebDriver:**

* HTTP server of the IE driver will listen for commands from language bindings. Defaults to 5555
* Specifies the IP address of the host adapter on which the HTTP server of the IE driver will listen for commands from language bindings. Defaults to 127.0.0.1
* Specifies the level at which logging messages are output. Valid values are FATAL, ERROR, WARN, INFO, DEBUG, and TRACE. Defaults to FATAL.
* Specifies the full path and file name of the log file. Defaults to stdout.
* Specifies the full path to the directory **used to extract supporting files** used by the server. Defaults to the TEMP directory if not specified.
* Suppresses diagnostic output when the server is started –silent
* **Mobile Test tools**
  + Ranorex Appium, Skill Mobile
  + Mantis is not a mobile test tool

### **What are the different types of Web Driver APIs supported in Selenium?**

|  |  |  |
| --- | --- | --- |
| **WebDriver Name** | **WebDriver API** | **Supported Browser** |
| Gecko Driver (a.k.a. Marinetto) | FirefoxDriver() | Firefox |
| Microsoft WebDriver (a.k.a. Edge) | InternetExplorerDriver() | IE |
| Google Chrome Driver | ChromeDriver() | Chrome |
| HTML Unit Driver | WebClient() | {Chrome, FF, IE} |
| OperaChromium Driver | ChromeDriver() | Opera |
| Safari Driver | SafariDriver() | Safari |
| Android Driver | AndroidDriver() | Android browser |
| ios Driver | IOSDriver() | ios browser |
| EventFiringWebDriver | EventFiringWebDriver() | ALL |

**Common way of Locator**



* What is the advantage of using the Selenium webdriver gem as Ruby client driver for Selecium RC?
  + Convenience methods for AJAX
  + Robust Rake take
  + Idiomatic interface to the Selenium API
* **Working with AJAX applications** do not have items for the test when tests get to command
* When browser and tests will run on the same machine then Selenium Server does not require
* What are the important factor responsible for using the Selenium-Server with Selenium-WebDriver?
  + Not using Java binding
  + Connect to a remote machine
  + Using selenium grid to distribute test
* How does network data capturing done in Selenium?
  + By using API in default Selenium class.