SELENIUM INTERVIEW QUESTIONS

What is automation and what are the benefits?

- Using software or computers to automate manually done actions.
- It helps you save money and improves efficiency.
- It is faster and safer since there will be no human error.
- You can run it anytime, anywhere, 24 hours.
- Test automation is automating the testing process using specific softwares. This requires new sets of skills and using tools like Selenium, cucumber, Serenity, UFT, appium, Protractor, etc.

What are the advantages of Selenium?

- Selenium is open source and free to use without any licensing cost
- It supports multiple languages like Java, Ruby, Python, C#...
- It supports multi-browser testing
- It has a good amount of resources and a supporting community.
- It supports many operating systems like Windows, Mac, Linux ...
- It interacts with the web applications

What are the limitations and disadvantages of Selenium?

- Selenium supports **only web-based applications**, does not support windows-based application
- No built-in reporting tool, it needs third party tools for report generation activity
- Cannot work with graphics, captchas, barcodes, shapes
- It does not support file upload facility.
- Hard to write good and stable locators since they are dependent on web developers.
- We have to use external libraries and tools for performing tasks like testing framework (TestNG, JUnit), reading from external files (Apache POI for excel)

Shortcomings of Selenium and solutions

- Mobile Applications:
  Selenium automated testing works well with desktop browsers, however, the stand-alone Selenium WebDriver cannot run on native mobile operating systems. To accomplish cross-browser testing including native mobile apps, testers need to use plugins like Selendroid or Appium.
- Parallel Testing:
  Testers can accomplish parallel testing using the Selenium Grid.
- Captchas/Graphics/Barcodes:
  There is no inherent capability in Selenium to solve captchas. Need 3rd party tools but still cannot achieve 100% automation.
- Pop-up Windows Handling:
  It's beyond selenium's capabilities. We could use AutoIT to handle the windows based popups.

How do you decide if tests can be automated?

- Functionality has to bring value to the business.
- Functionality needs to be working (no bugs). Whatever you want to automate needs to be working without any issue so you can automate it.
- Functionality must be stable.

What functionalities cannot be automated using Selenium?

- Captcha
- Broken functionalities: if the function is not working, you cannot automate it.
- QR or Bar Codes can only be automated by using 3rd party API.
- 3rd party dependencies: For example, on 4stay.com when you search for hotel, it brings map view which is actually coming from GoogleMaps so if you automate it, when there is a change done on the map by Google we will not be able to handle it since it is a 3rd party application.
- Test cases related to cosmetics, look, etc., because they are not about functionality.

What types of testing can you automate with Selenium?

- Functional Tests (positive/negative, UI)
- Smoke Tests
- Regression Tests
- Integration Tests
- End-to-end Testing
- Data Driven

What are the test types we don't automate with Selenium?

- Performance, stress, load testing: Selenium is about functionality only.
- Manual ad hoc testing
- Usability testing ???
- Pure DataBase testing. If you do database only, you don't need Selenium.
- Unit testing
- Static testing

What is a locator and what are the locator types?

- Locator is a way to find the elements in selenium. Locators help us to interact with the browser actions; such as, clicking on the buttons, selecting an option in the dropdown menu, etc.
- There are 8 locators:
    - Id
    - Name
    - className
    - tagName
    - linkText
    - partialLinkText
    - XPath
    - CSS selector

What are the Selenium Tools / Components?

- Selenium **IDE** is implemented as a Chrome and Firefox extension, and allows you to record, edit, and debug tests.
- Selenium **RC** is to write automated web application UI tests in any programming language
- Selenium **WebDriver** executes your tests against different browsers
- Selenium **GRID** runs your tests on different machines against different browsers in parallel.

How do you handle dynamic elements?

- Find the static part of the id and write a locator(**xpath or css**) and then use **startswith**, **contains**, **endswith**
  - contains( ) : //*[contains(@name=`btn`)]
  - startwith( ) : //label[startswith(@id, `message`)]
  - text( ) : //td[text() = `usedId`]
  - or & and : //input[@type = `submit` AND @name = `login`]
- Using parent, child, sibling relationship starting from the stable/static element.
- I have utility methods that work with tables. I have a method that takes a table webelement and returns all the column names. I have a method that takes a table, number and returns all the data in that row.

How to find element that have dynamic attributes?

I can find element with dynamic attributes using xpath or css

```
<input id = '123abc456'>
```

If the first part of the attribute is stable

xpath //input[starts-with(@id, '123')]
css input[id^='123']

If the last part of the attribute is stable

xpath //input[ends-with(@id, '456')]
css input[id$='456']

If searching by any substring of the attribute value

xpath //input[contains(@id, 'abc')]
css input[id*='456']

How to find elements in a table?

First Name	**Last Name**
Ali	Durhan
Mehmet	Erdogan

We have a table with three rows that have tags **tr.** every cell is **td.**
I can find elements in the table using different ways. For example we can use **index.**

This is how we find **Durhan** on the given table:

//table/tr[2]/td[2]

We can also use parent-child relationships. Let's find the row that contains the name Ali. This xpath finds the cell that has text Ali then finds its parent using /..

//td[.= 'Ali']/..

Difference Between close() and quit() command?

- driver.close() used to close the current browser
- driver.quit() used to close all the browser instances

What is Xpath?

- Xpath is used to find the location of any element on a webpage using html structure.
- We can navigate through elements and attributes in an XML document to locate webElements such as textbox, buttons, checkbox, etc. in web Page

How Can We Move To The Parent Element Using Xpath?

- Using the **(::)** expression in xpath, we can move to the parent element from the child but this can only be done with XPath, NOT with CSS.

How can we move to the nth child element using Xpath?

- There are two ways:
    - using square brackets with index position
      For ex: **div[2]** will find the second div element
    - using position ( ) method
      For ex: **div[position()=2]** will find the second div element

What is the difference between “relative” and “absolute” XPath?

Absolute XPath:

- Parent/child path starts with **(/)** single slash showing the root element and goes all the way to the element we’re locating.
- If there are any changes in the path of the element then the XPath will fail.
- Not suggested to use as the developers may change the path.

Relative Path:

- The path starts from the middle of the HTML DOM structure with double slash **(//).**
- Starts with // which means it searches the element anywhere at the webpage.
- No need to write a long XPath. More preferred!

Difference Between Xpath And Css Selector?

- Xpath can search elements backward or forward while css works only in forward direction.
- Xpath can work with text, css cannot work.
- Xpath has more combinations and can search by index, css cannot search by index, but css is working faster than xpath.
- CSS is faster than XPath but I never experienced it in my framework as it is a smaller project.

Implicit / Explicit / Fluent Wait

- **Implicit Wait:**

    - Wait time is applied to all elements in the scripts. It is more global.
    - We don't specify the expected condition on the element to be located.
    - It is recommended to use when the elements are located with the time frame specified in the implicit wait.
    - If the wait time set for 10 seconds but the element found in 3, the remaining 7 seconds are not waited in both implicit and explicit wait
- **Explicit Wait:**
    - Wait time is applied ONLY to those elements which are intended / specified by us.
    - We need to specify the "ExpectedCondition" on the elemen to be located.
    - It is recommended to use when the elements are taking a long time to load and also verifying the property of the element like visibilityOfElement, elementToBeClickable, elementToBeSelected, etc.
- **Syntax:**
    - *driver.manage().timeouts().**implicitlyWait**(5,TimeUnit.SECONDS);*
    - *webDriverWait wait = new WebDriverWait (driver, 5);*
      ***wait.until** (ExpectedConditions.visibilityOf(element);*
- **Fluent Wait:**
    - It is used to tell the web driver for a condition as well as the frequency with which we want to check the condition before throwing an "ElementNotVisibleException"
    - Frequency: it is setting up a repeat cycle with the time frame to verify/check the condition at the regular interval of time.
    - I know fluentWait but we never used it in our project. I use implicit or explicit wait so far.

What is "Thread.sleep()"?

- It is part of Java, not Selenium. It holds the execution of the code for the given time.
- Unlike, Implicit or Explicit wait, it will wait for 10 (given time) seconds even if the element is found or loaded in 3 seconds / earlier.
- It is not suggested to use Thread.sleep().
- It throws an exception so you must handle or throw it.

Why and how to locate elements using "text"?

- There might be cases where no locator is available to find an element so you may need to use text().
- For example, you're trying to find the "announcement" button but there is no id, class, etc. available so you can just search for the text "announcement" which is written on the button. In this case, Selenium will look for the text to locate it.
- It is not that much recommended to use text() since the developers change it a lot.
- Syntax:
    - Matching exact text: //tag[.='text'] → *//span[.= 'announcement']*
    - Matching partial text: //tag[contains(text(), 'text')] → *//span[contains(text() ,'announcement')]*

How to handle static drop down menus?

- To be able to select an element/option from a dropdown menu, we need to create a **new object out of select class** which has already been blueprinted by Selenium developers.
- Import Select Class and create an object. This new object will come with its own methods like selectByValue, selectByIndex, etc.

  Methods from Select Class:

  Select s = new Select(driver.findElement(By.id("cat")));
  Select by Value = *s. selectByValue("5");*
  Select by VisibileText = *s. selectByVisibleText("Day");*
  Select by Index = *s. selectByIndex(3);*

- If there is a select tagName, it means that 99% it is a static dropdown.

How to handle drop down menus without SELECT tag?

- If the dropdown list has no select tag, we cannot use the select class's methods so we will need to locate and click on the options ourselves.
- We need to treat each option as a separate web element and find them to interact with them accordingly.
- To click on an option under the dropdown:
  - Find the dropdown list and click on it
  - Find and click on the option you want to select

How to handle Dynamic Dropdown in Selenium?

- When the options on the dropdown list are loaded based on the selection or action by the user, we call these dropdown lists **dynamic** since the options are changing, not static.
- For example, when you search for a driver's name in the dropdown list by his last name, you type "dur" and all names including "dur" will be listed and you can pick any of them. BUT when you try to find the element using XPath ***//a[@value = 'dur']*** there will be no match!!!
- You will need to click on the dropdown first **and then** look for the element since it will be visible now!
- Since there might be more than one option including "dur", selenium will pick the first option by default. If you want to pick the second one, you will need to tell Selenium to do so by explicitly defining it: ***(//a[@value = 'dur']) [2]***

Html Alerts - JavaScript Alerts - OS Windows Alerts

- **Html Alerts:**
  They are like regular web elements so we can locate and click on them.
- **JavaScript Alerts:**
  Alert is a small message box which displays on-screen notification to give the user some kind of information or ask for permission to perform a certain kind of operation. It may also be used for warning purposes. Selenium can only work with web based codes. If a pop-up window is a Java Alert, Selenium cannot handle it with driver reference. You need to create an Alert object and use .switchTo() method to be able to take any action on the Java Alert. We can also call them Browser Alerts.
  Alert alert = driver.switchTo().alert();
  *driver.switchTo().alert().**accept**()* = to give a positive response: OK, Done, etc.!

*driver.switchTo().alert().**dismiss**()* = to give a negative response: cancel, deny, etc.!
*driver.switchTo().alert().**getText**()* = to take text.
*driver.switchTo().alert().**sendKeys**()* = to add text and then click on the button.

- **Windows/OS Alerts:**
  Selenium cannot handle OS (operation system)/windows based alerts. There are 3rd party tools such as AutoIT, Robot, etc.

What is the syntax for switching windows?

- Selenium cannot directly locate the window you want to handle. You need to store all window handles and loop through them.
- I have this in my BrowserUtils class:

```
public static void switchToWindow(String targetTitle) {
  String origin = Driver.getDriver().getWindowHandle();
  for (String handle : Driver.getDriver().getWindowHandles()) {
    Driver.getDriver().switchTo().window(handle);
    if (Driver.getDriver().getTitle().equals(targetTitle)) {
      return;   }}
  Driver.getDriver().switchTo().window(origin);          }
```

- driver.getWindowHandle(); = This will handle the current window that uniquely identifies it with the driver instance.
  driver.getWindowHandles(); = To handle all opened windows.
  driver.getCurrentWindowHandle(); = Returns the current window handle.

How to handle FRAMES in Selenium?

- IFrame is a web page which is embedded in another web page or an HTML document embedded inside another HTML document.
- The IFrame is usually used to insert content from another source, such as an advertisement into a web page.
- Selenium cannot detect the frames by just using the page or inspecting it. Selenium thinks that the frame is part of the current page so it reacts accordingly.
- It is impossible to click iFrame directly through XPath since it is an iFrame hosted on the current page. First, you need to SWITCH TO the frame and then you can click using XPath.
- Syntax: Use one from the three options to switch: id, web element, and index

```
driver.switchTo().frame("a077aa5e");  =  this is with the "id"
driver.switchTo().frame("webElement");
driver.switchTo().frame("index");
```

- driver.switchTo().defaultContent(); = *when you're done working with the frames, you HAVE TO switch back to the default page to be able to control it.*
- *driver.switchTo().parentFrame()* = *goes to previous frame (only one level up)*
- This is how you can find the number of iFrames on a webpage:
  int numOfFrames=driver.findElements(By.tagName("iframe")).size();

What operations can you do using the ACTIONS class?

- *Thanks to Actions class in Selenium, you can have special keyboard and mouse interactions. You can drag & drop, double click, scroll down, right click, etc.*
- *To be able to have control of the Action class, you need to create an object of it with the driver.*
- Syntax: *you need to build and perform at the end.*

```
Actions a = new Actions (driver);
a.moveToElement(webElement).build().perform();
```

- Methods:

.clickAndHold()	.dragAndDrop(*source*, *target*)
.contextClick()	.keyDown(modifier_key)
.doubleClick()	.moveToElement(*toElement*)
.release()	

How do you UPLOAD a file in Selenium?

- First, **locate** the element which takes the path of the file we want to upload. It is usually "Choose File" button

```
WebElement input = driver.findElement("id");
```

- Second, **provide** the path to the file using the ***sendKeys*** method:

```
input.sendKeys("/path/to/file/here");
```

How do you do "Cross Browser" Testing?

- *There are multiple ways to achieve cross browser testing depending on the framework. TestNG does it differently even with and without XML; or Cucumber and JUnit, each of them has their own way to do it.*
- In my framework, I use JUnit/Cucumber. To handle different browsers, I use a properties file where I can pass the name of the browser that I want to work with. I also have a Driver utility class that can open different types of browsers depending on the value in the properties file. I have a switch statement in the Driver utility so if I define the value of browser as "chrome" in the properties file, Driver utility class will return a ChromeDriver.
- I can also provide the value of the browser from the terminal when I run tests using maven:

```
mvn test -DBrowser="firefox"        or
mvn test -DBrowser="chrome"
```

What version of Selenium do you use right now?

- JDK (JAVA) - 1.8
- IntelliJ - 2018.03.04
- Selenium - 3.141.59
- TestNG - 6.14.3
- Cucumber – 4.2.6
- Maven - 3.6.0
- GIT - 2.17.2

How do you do "Headless Browser Testing"?

- A headless browser is a type of software that can access web pages but does not show them to the user and can pipe the content of the web pages to another program. Unlike a

normal browser, nothing will appear on the screen when you start up a headless browser, since the programs run at the backend. This is faster.

- In my project, I include headless browsers in the Driver Util class.

```
switch (browser) {
        case "chromeHeadless":
          WebDriverManager.chromedriver().setup();
          driver = new ChromeDriver(new ChromeOptions().setHeadless(true));
          break;
        case "firefoxHeadless":
          WebDriverManager.firefoxdriver().setup();
          driver = new FirefoxDriver(new FirefoxOptions().setHeadless(true));
          Break;
        case "PhantomJS": // this is a 3rd party tool
          WebDriverManager.phantomsjsdriver().setup();
          driver = new PhantomJS();
          break;
        case "htmlUnitDriver": // this is from Selenium
          driver = new HtmlUnitDriver();
          break;
        default:
          throw new RuntimeException("Illegal browser type!");    }
```

How can you find the number of the checkboxes and checked boxes?

- Find the common attribute for all of the checkboxes.
- Store all of the checkboxes (each of them are separate objects) in one single **list object /element** by using **findElements()**.
- Use **.size()** method in the findElements().size() method. *// this is for the size*
- Create a for-loop and put **count++** in it.
- Use **isSelected()** method with an if statement and if it gives **"true"** it will increment the value of count.
- Print the count after the loop.

What are the potential reasons if you cannot find an element?

- Locator might have been changed by the developer.
- It can be an iFrame
- Waiting time:: page is loading slowly or Element is dynamic:: locator
- Page is not fully loaded/opened
- Page changes and that element does not exist anymore

How do you handle a dynamic pop-up that comes randomly?

- I can use a try/catch block with alert

What do you do when you get a NoSuchElementException?

- I check if the locator is correct.
- I make sure I don't have any synchronization issues.

- I double-check if the element is hidden inside an iframe.

What is the difference between driver.get() and driver.navigateTo() ?

- driver.get() is to open an URL and it will wait till the whole page gets loaded
- driver.navigateTo() is to navigate to an URL and it will not wait till the whole page get loaded

findElement vs findElements?

- FindElement():
  This method returns the first WebElement and gives Exception if the element is not found. It will return only ONE element.
- FindElements():
  This method returns **List** <WebElement> but does not give Exception if the element(s) is not found as a result list has **null** values.

What Exceptions do you usually face in Selenium?

- **NoSuchElementException**
  I usually face this exception which usually happens because of synchronization issues, incorrect locators, or hidden elements in iFrame.
- **StaleElementException**
  It usually happens when the element has been deleted entirely or no longer attached to the DOM. This is how I handle it:
  - Element is not attached to DOM: 'try catch block' within 'for loop' or
  - 1.Refresh the page and try again for the same element.
  2.Wait for the element till it gets available.
- **TimeOutException**

Difference between isDisplayed(), isEnabled(). and isSelected() method in selenium WebDriver?

- **isDisplayed():**
  verifies the presence of a web element within the web page. It checks for the presence of all kinds of web elements available
  if found = true if not found = false
- **isEnabled():**
  verifies if the web element is enabled or disabled within the web page. It is primarily used with buttons.
- **isSelected():**
  verifies if the web element is selected or not. It is used with radio buttons, dropdowns and checkboxes.

Why do we need to handle cookies?

- Each cookie is associated with a name, value, domain, path, expiry, and the status of whether it is secure or not. In order to validate a client, a server parses all of these values in a cookie.

- When Testing a web application using a selenium web driver, we may need to create, update or delete a cookie.
- For example, when automating Online Shopping Application, we may need to automate test scenarios like place order, View Cart, Payment Information, order confirmation, etc.
- If cookies are not stored, we will need to perform login action every time before we execute above listed test scenarios. This will increase your coding effort and execution time.
- The solution is to store cookies in a File. Later, retrieve the values of cookie from this file and add to it your current browser.
- session. As a result, you can skip the login steps in every Test Case because your driver session has this information in it.
- The application server now treats your browser session as authenticated and directly takes you to your requested URL.

Do you use JavaScriptExecutor?

- This helps me write my own JavaScript. JS has way more control than selenium.
- We can send JS commands to the browser with using this class
  JavaScriptExecutor jsExecutor=(JavaScriptExecutor)driver;
  executeScript(); performs the command
  Inside the parameter is where you put JS code
- jsExecutor.executeScript("alert('WARNING: This is a useless message');" This code will bring up a JS popup
- You can also put 2 parameters is .executeScript("js code",element);
  Used for scrolling (selenium is not good with scrolling, you can say a challenge is when I was working on a terms and conditions page, where you have to read the page before clicking on continue.
  When I tried using selenium and actions class it didn't work, so I used javaexecutor) and clicking an element;

File Download And Upload

- **Download:**
  - Selenium itself cannot verify file downloads, can click on download link but can't go outside the browser and open the downloaded file
  - Other tools need to be used for that Robot and AutoIT
- **Upload:**
  Selenium handles the upload, but does it differently compared to the actual user.
  Steps
  - Find the element that triggers the upload window
  - Find the path of the file you want to upload

  Store into a String
  Ex: String → file= *"C:\\Users\\Andy\\Desktop\\folder1\\file.key";*
  Then *driver.findElement(upload button).sendkeys(file);*

What is SSL Certification and How do you resolve certification issues?

- An **SSL** (Secure Sockets Layer) certificate is a digital certificate that authenticates the identity of a website and encrypts information sent to the server using **SSL** technology.
- **Case:**
  You might receive a notification window stating that your website is not secure and to be able to complete your automation, you will need to hit the "**proceed anyway**" button.
- There are different certifications and each may have different ways of accepting and continuing the process. You can handle this with DesiredCapabilities or ChromeOptions classes.
- **Syntax:** *(for general chrome profile)*

```
DesiredCapabilities ch = DesiredCapabilities.chrome();
ch.setCapability(CapabilityType.ACCEPT_INSECURE_CERTS, true);
ch.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);

ChromeOptions c = new ChromeOptions();
c.merge(ch);     // this will merge the capability into ChromeOptions.

System.setProperty("webdriver.chromeDriver", " ");
WebDriver driver = new ChromeDriver(c);
        // this is how we pass capabilities into the "driver so it accept all
        certifications when they appear on the screen.
```

SELENIUM SYNTAX QUESTIONS

How to maximize web pages?

➔ driver.manage().window().maximize();

In some cases, maximize() will not work>so what will be the way around?

➔ ChromeOptions options = new ChromeOptions();
options.addArguments("*startmaximized*");

How To Handle Multiple Windows/Tabs?

➔ for(string handle: driver.getWindowHandles()){
driver.switchTo().Window(handle)
if(driver.getTitle().equals(targetTitle){
break; } }

How To Find All Links In The Page?

➔ List<WebElement> list = driver.findElements(By.tagName("a"));

How to Drag and Drop?

➔ *We need to use Actions class*

Actions action = new **Actions**(**driver**);
action.**clickAndHold**(driver.findElement(By.id("item")))
.**moveToElement**(driver.findElement(By.id("destination")))
.**release**().*build*().*perform*();

How to Scroll Down?

➔ *We need to use JavaScript Executor.*

WebDriver driver = new ChromeDriver();
JavascriptExecutor jse = (JavascriptExecutor)driver;
jse.executeScript("window.scrollBy(0,250)", "");
or
jse.executeScript("scroll(0, 250);");

How to Scroll Up?

➔ jse.executeScript("window.scrollBy(0,-250)", "");
or
jse.executeScript("scroll(0,-250);");

How to Handle 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
```

How to check if an element is present/visible/enable/and to check text present?

➔ **To check Element Present:**

```
if(driver.findElements(By.xpath("value")).size() != 0){
        System.out.println("Element is Present");
}else{
        System.out.println("Element is Absent");}
```

or

```
if(driver.findElement(By.xpath("value"))!= null){
        System.out.println("Element is Present");
}else{
        System.out.println("Element is Absent"); }
```

➔ **To check Visible:**

```
if(driver.findElement(By.cssSelector("a > font")).isDisplayed()){
        System.out.println("Element is Visible");
}else{
        System.out.println("Element is InVisible"); }
```

➔ **To check Enable:**

```
if(driver.findElement(By.cssSelector("a > font")).isEnabled()){
        System.out.println("Element is Enable");
}else{
        System.out.println("Element is Disabled"); }
```

➔ **To check text present:**

```
if(driver.getPageSource().contains("Text to check")){
        System.out.println("Text is present");
}else{
        System.out.println("Text is absent"); }
```

How to check the multiple selected values in dropdown?

➔ Select carsList = new Select(el)
carList.getSelectedOptions(): *//returns the the selected options a list (List<webelement>)*
for each : carList.getSelectedOptions()

What is the syntax for double click action ?

To perform any actions against web element using actions class, we need to locate the element first:

WebElement el = driver.findELement

Actions actions = new **Actions** (driver).perform actions.doubleClick(el).perform() ;
actions.moveTo(el).perform actions.doubleClick.perform
actions.moveTo(el).doubleClick().build.perform();

How to check the selected value in dropdown?

Select carsList = new Select(**el**);
carList.getFirstSelectedOption();
assertequals("some text", **carList.getFirstSelectedOption().getText()**);

What Is The Syntax For Uploading A File?

public void fileUpload(String path){

WebELement upload = driver.findELement; Upload.sendKeys(path) }

- *We need to locate the upload button in html.*
- *The element will have tag input.*
- *Then we do sendKeys by passing the path to the file which we want to upload.*

Creating a dynamic path

➔ *Building a dynamic path for a file inside our project Path to the project location:*

String projectDir= System.getProperty("user.dir") // project directory
String file= "src/test/resources/test_data/myfile.txt";
Element.sendKeys(projectDir+file);

What Is The Syntax For Switching Windows?

```
public static void switchToWindow(String targetTitle) {
        String origin = Driver.getDriver().getWindowHandle();
        for (String handle : Driver.getDriver().getWindowHandles()) {
                Driver.getDriver().switchTo().window(handle);
                if (Driver.getDriver().getTitle().equals(targetTitle)) {
                        return; } }
        Driver.getDriver().switchTo().window(origin); }
```

- *This method is from the BrowserUtil Class*
- *It switches to new window by the exact title*

How to input text in the text box without calling the sendKeys()?

```
   //Use                                                                javascriptExecutor
JavascriptExecutor JS = (JavascriptExecutor)webdriver;
   //To                                        enter                                  username
JS.executeScript("document.getElementById('User').value= 'www.google.com'");
   //To enter password
JS.executeScript("document.getElementById('pass').value=' tester'");
```

How to press ENTER key on text box in Selenium WebDriver?

➔ Driver.findElement(By.xpath("xpath")).sendKeys(Keys.ENTER);

How would you verify the position of the Web Element on the page?

➔ *WebElement class has a get Location method with returns the top left corner of the element*
element.getLocation();

How To resize browser Window Using Selenium WebDriver?

➔ *To resize the browser window to particular dimensions, we use 'Dimension' class to resize the browser window.*

Create object of Dimensions class

```
Dimension d = new Dimension(480,620);
```

Resize the current window to the given dimension

```
driver.manage().window().setSize(d);
```

How do you find a text in a webpage?

//tagname[contains(text(), 'text')]	:	*contains certain test*
//tagname[.= 'text']	:	*contains exact text sometimes doesn't work Selenium*

How to get all the preceding siblings of Apple?

➔ *This will give "Samsung Mobiles"*

Xpath: "//ul/li[contains(text(), 'Apple Mobiles')]/**preceding-sibling**::li"

How to get all the following siblings of Apple?

➔ *This will give all the following siblings (Nokia Mobiles, HTC Mobiles, Sony Mobiles, Micromax mobiles)*

Xpath: "//ul/li[contains(text(),'Apple Mobiles')]/**following-sibling**::li"

Selenium Grid & Source Lab & AWS

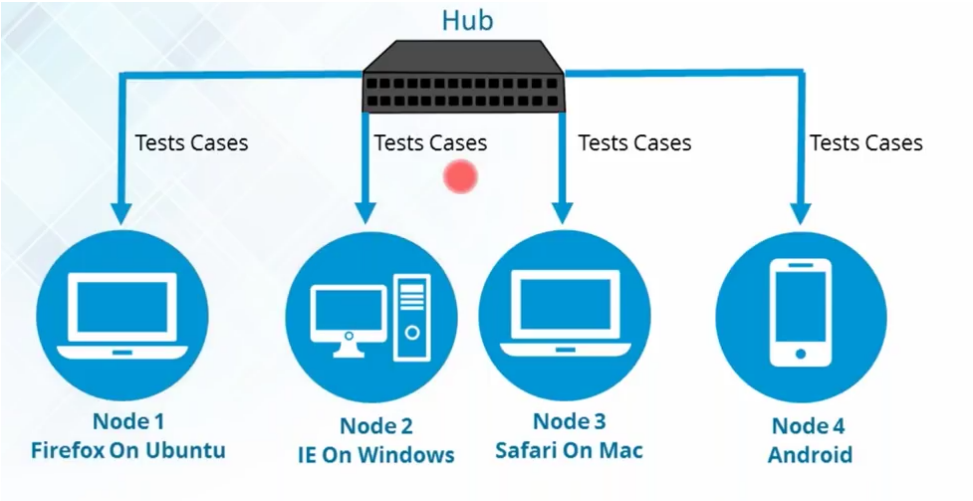
What is Selenium Grid?

- Selenium Grid gives you the ability to run your automated tests in **different browsers**(and their different versions) and **platforms** (basically Operating systems and their versions: Window, Linux, Mac) VISGRID
- Selenium Grid is very useful if you have many tests (500+) as you can save from time.
- Instead of doing
    - WebDriver driver=new Chromedriver()
- Do
    - WebDriver driver=new RemoteWebDriver(url,capabilities)
        //Contains 2 parameters in constructors

When do you use Selenium Grid?

- Selenium Grid can be used to execute same or different test scripts on **multiple platforms and browsers** concurrently so as to achieve distributed test execution.

How is the Architecture of Selenium Grid?



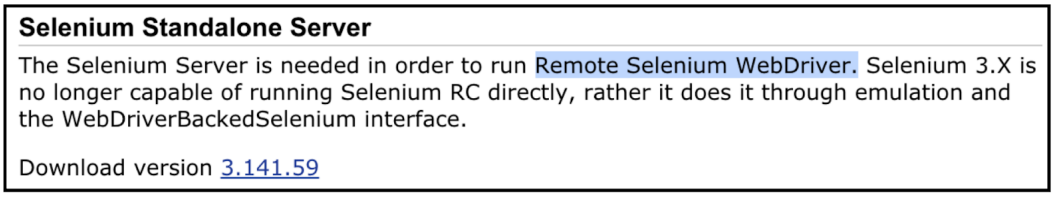
- **HUB**:
    - The hub is the central point that will receive all the **test requests** and **distribute** them to the right nodes.
    - There should be only **one hub** in a grid.
    - The machine containing the hub is where the tests will be triggered, but you will see the browser being automated on the node.
    - The hub can be **parametrized** with a **json** file.
- **NODE:**
    - Nodes are the **Selenium instances** that will **execute the tests** that you loaded on the hub.
    - Nodes can be launched on multiple machines with **different platforms** and **browsers**.
    - Nodes can be **parametrized** with a **json** file.

How does Selenium Grid work?

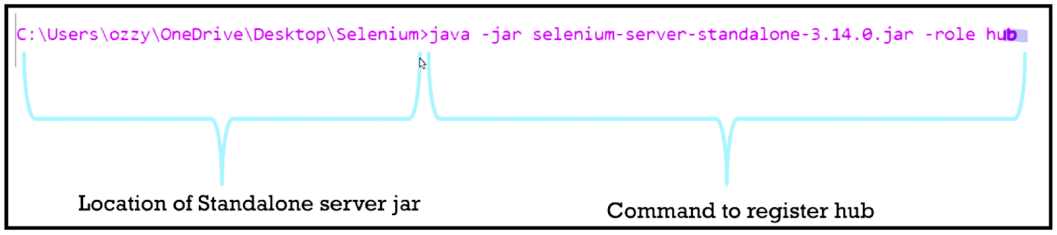
- Grid is a set up that consists of **Hub** and **Nodes**. Both are started using the selenium-server.jar executable.
- There is a **main machine** which is called a **Hub** and multiple **nodes** (the machines that actually run your tests)
- **The order of execution is:**
  Your code → Remote driver → Selenium Hub → Selenium Nodes *(might be multiple ones)*
  - You ask your Selenium Hub to run your tests
  - Then selenium hub will find a node that is linked to the hub and run your test from there
  - You can have as many nodes as you want but only one hub

How do you set up your selenium grid?

1. Download Selenium standalone server jar: https://www.seleniumhq.org/download/



2. Register your hub by executing the below code in the command prompt:
   java -jar selenium-server-standalone-<version>.jar -role hub



3. Verify if hub is registered:
   The hub will automatically start-up using port 4444 by default. To change the default port, you can add the optional parameter -port when you run the command. You can view the status of the hub by opening a browser window and navigating to:http://localhost:4444/grid/console

```
11:53:02.319 INFO [Hub.start] - Selenium Grid hub is up and running
11:53:02.320 INFO [Hub.start] - Nodes should register to http://192.168.2.21:4444/grid/register/
11:53:02.321 INFO [Hub.start] - Clients should connect to http://192.168.2.21:4444/wd/hub
```

   Open a browser and go to http://localhost:4444/grid/console
   (below is the Grid Console showing Selenium is up!)

4. Create / Run a node:

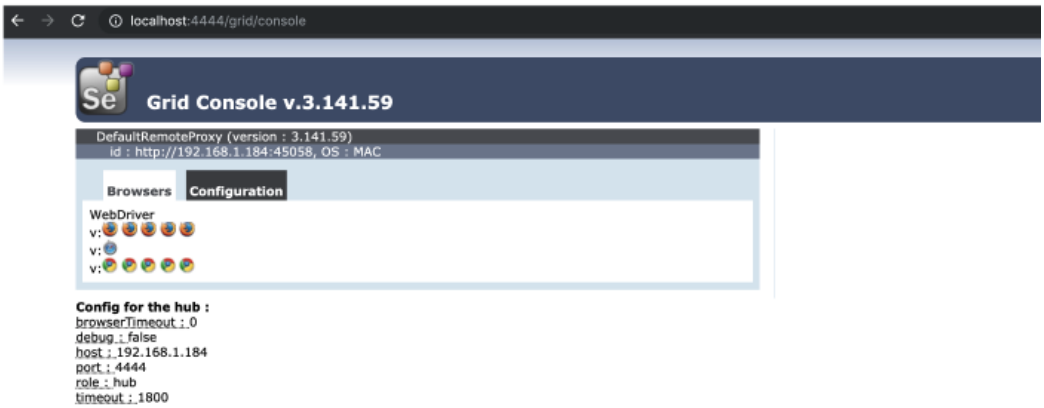
```
C:\Users\ozzy\OneDrive\Desktop\Selenium>java -Dwebdriver.chrome.driver="C:\Users\ozzy\OneDrive\Desktop\Selenium\chromedriver.exe" -jar selenium-server-standalone-3.14.0.jar
 -role webdriver -hub http://192.168.2.21:4444/grid/register -port 5566
```

   1. C:\Users\ozzy\OneDrive\Desktop\Selenium>
   2. java Dwebdriver.chrome.driver="C:\Users\ozzy\OneDrive\Desktop\Selenium\chromedriver.exe"
   3. jar seleniumserverstandalone3.14.0.jar
   4. role webdriver
   5. hub http://192.168.2.21:4444/grid/register
   6. port 5566

5. Verify if node is created successfully and Check Grid console:

```
14:51:01.780 INFO [GridLauncherV3$3.launch] - Selenium Grid node is up and ready to register to the hub
14:51:01.921 INFO [SelfRegisteringRemote$1.run] - Starting auto registration thread. Will try to register every 5000 ms.
14:51:01.921 INFO [SelfRegisteringRemote.registerToHub] - Registering the node to the hub: http://192.168.2.21:4444/grid/register
14:51:02.600 INFO [SelfRegisteringRemote.registerToHub] - The node is registered to the hub and ready to use
```

   Go to localhost:4444/grid/console in order to view your selenium grid console:



6. Configure Driver Class:

   Create a **RemoteWebDriver** instance.
   Provide **hub URL** address and **DesiredCapabilities** in order to specify node preferences.

```
case "remote_chrome":
    try {
        WebDriverManager.chromedriver().setup();
        ChromeOptions chromeOptions = new ChromeOptions();
        chromeOptions.setCapability( capabilityName: "platform", Platform.ANY);
        driverPool.set(new RemoteWebDriver(new URL( spec: "http:localhost:4444/wd/hub"), chromeOptions));
    } catch (Exception e) {
        e.printStackTrace();
    }
    break;
case "remote_firefox":
    try {
        WebDriverManager.firefoxdriver().setup();
        FirefoxOptions firefoxOptions = new FirefoxOptions();
        firefoxOptions.setCapability( capabilityName: "platform", Platform.ANY);
        driverPool.set(new RemoteWebDriver(new URL( spec: "http:localhost:4444/wd/hub"), firefoxOptions));
    } catch (Exception e) {
        e.printStackTrace();
    }
    break;
```

How to configure the nodes?

Configure the nodes by command line:

- 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.
- To change this and other browser settings, you can pass in parameters to each -browser switch (each switch represents a node based on your parameters). If you use the -browser parameter, the default browsers will be ignored and only what you specify on the command line will be used.
- -browser browserName=firefox,version=3.6,maxInstances=5,platform=LINUX

Configure the nodes by JSON:

- java -Dwebdriver.chrome.driver="chromedriver" jar selenium-server-standalone-<version>.jar -role node -hub http://localhost:4444/grid/register -nodeConfig nodeconfig.json

How do you run your tests in multiple threads parallely?

Thread is like one process or instance of application run and there are 4ways for parallel testing:

1. We can create multiple cukes runner with different tags
    a. Ex. CukesRunner has tag “@Test”
    b. Smokerunner has @smoke
    c. Regression runner has @Regression
2. Create testng xml and add those runner class under one test
    a. Add all 3 runners in one xml
    b. Then next to verbose=2(on top of xml file) add “parallel=”classes” thread-count=”10”>
3. Then make sure our driver opens a remote WebDriver that is point to hub
    a. Add the code in driver class
4. Run the testng xml by itself or using maven.

How do you automate multi browser testing?

Change the browser to something else in my properties file in my framework

○ When I want to run my tests in different browser

○ Ex; “Browser=chrome” to “Internet Explorer”

○ This method works if your tests are less than 500 tests

● In my framework, I implemented Selenium Grid and I can run tests in different cloud machines using different browsers

AWS

What is AWS?

- AWS is providing cloud VM. Create an EC2 instance.
- I can use this instance with a remote desktop. Actually, after launching my instance I just use it like a regular computer.

Do you work with AWS?

- I am working with EC2 instances.
- Basically, that is my virtual machine.
- When I have Selenium Grid, I have different virtual machines and each machine I am running separately.
- For instance, to minimize the time for regression tests, it is really efficient, it saves a lot of time to our company

What is the base page?

- We store our common functionalities in a base class and later we extend that base class and use it in other classes.

:::::Parallel testing:::::

To perform parallel testing, we need to configure the webdriver properly. If we are using a singleton driver, the only way to do parallel testing is to use forks. Fork, it's one JVM process.##Forks require a lot of cpu and ram, that's why it's not recommended to use. In our project, we use ThreadLocal driver.ThreadLocal allows to create a copy of the webdriver object during the run time. For every feature file that we run, threadlocal will create the webdriver copy. Also, we are using maven-surefire plugin where we can specify maximum number of threads:<parallel>methods</parallel><perCoreThreadCount>false</perCoreThreadCount><threadCountMethods>4</threadCountMethods>1. webdriver = test scenario = 1 browser

ThreadLocal class allows us to avoid forks.