50 Selenium interview question

This is to the reference with our LinkedIn communication, kindly find the first 50 set of attached **Selenium interview question.**

Question 1: What is the main difference between selenium 2.0 and 3.0?

Answer 1: In Selenium 2.0,

Selenium commands directly interact with browser API. Browser API will interact with browsers via DOM structure after receiving commands from client. That's why for each kind of browser they provided browser specific and supportive methods, by default browser in selenium 2.0 is Firefox

In Selenium 3.0

In Selenium 3.0 has become a W3C standard (World wide web consortium), Selenium 3 would be majorly looking to be a choice of software testing tool for both web and mobile-based applications

Question 2: What are the different types of web driver APIs supported in selenium?

Answer 2: Following are the web drivers supported in Selenium.

	WebDriver Name	WebDriver API	Supported Browser
1	Gecko Driver (Marinetto)	FirefoxDriver()	Firefox
2	Microsoft WebDriver (Edge)	InternetExplorerDriver()	IE
3	Google Chrome Driver	ChromeDriver()	Chrome
4	HTML Unit Driver	WebClient()	{Chrome, FF, IE}
5	OperaChromium Driver	ChromeDriver()	Opera
6	Safari Driver	SafariDriver()	Safari
7	Android Driver, AndroidDriver()	Android browser	
8	Ios Driver	IOSDriver()	ios browser
9	EventFiringWebDriver	EventFiringWebDriver()	AL**L**

Question 3: Which of The WebDriver APIs Is The Fastest, And Why

Answer 3:

It is none other than the HTMLUnitDriver, which is faster than all of its counterparts.

The technical reason is that the HTMLUnitDriver doesn't execute in the browser. It employs a simple HTTP request-response mechanism for test case execution.

This method is far quicker than starting a browser to carry out test execution.

Question 4: What Do You Know About Selenium Grid?

Answer 4:

Selenium Grid is a tool which can distribute tests across multiple browsers or different machines. It enables parallel execution of the test cases. Using this, we can configure to run thousands of test cases concurrently on separate devices or browsers.

Question 5: Why Do You Use Selenium Grid?

Selenium Grid allows the test cases to run on multiple platforms and browsers simultaneously, and hence, supports distributed testing. This ability to parallelize the testing is what makes us use the Selenium Grid.

Question 6: What Are the Pros/Benefits Of Using Selenium Grid?

There is a no. of benefits of using the Selenium Grid.

- Support concurrent test execution and hence saves us a lot of our time.
- It presents us with the ability to execute test cases in different browsers.
- After creating multi-machine nodes, we can use it to distribute tests and execute them.

Question 7: What Is the Difference Between MaxSessions Vs. MaxInstances Properties of Selenium Grid?

Sometimes we get confused while differentiating between the MaxSessions vs. MaxInstances. Let's understand the difference with clarity.

1. MaxInstances: It is the no. of browser instances (of the same versions) that can run on the remote machine. Check the below commands.

-browser browserName=firefox,version=59,maxInstances=3,platform=WINDOWS

-browser

browserName=InternetExplorer,version=11,maxInstances=3,platform=WINDOWS

It means we can run three instances of both Firefox and IE at the same time. So, a total of six different browsers (FF & IE) could run in parallel.

2. MaxSession: It dictates how many browsers (independent of the type & version) can run concurrently on the remote machine. It supersedes the "MaxInstances" setting.

In the previous example: If the value of "maxSession" is one, then no more than a single browser would run. While its value is two, then any of these combinations (2FF, 2IE, 1FF+1IE) can run at a time.

Question 8: What are The Locators Selenium Supports?

Following is the list of supported locators by Selenium.

- ID: Unique for every web element
- Name: Same as ID although it is not unique
- CSS Selector: Works on element tags and attributes
- XPath: Searches elements in the DOM, Reliable but slow
- Class name: Uses the class name attribute
- TagName: Uses HTML tags to locate web elements
- LinkText: Uses anchor text to locate web elements
- Partial Link Text: Uses partial link text to find web elements

Question 9: Which of The Id, Name, XPath, Or CSS Selector Should You Use?

If the page has unique names or identifiers available, then we should use **ID**.

If they are not available, then go for a CSS selector as it is faster than the XPath.

When none of the preferred locators is present, then you may try the XPath.

Question 10: What Does A Single Slash "/" Mean In XPath?

A single (forward) slash "/" represents the absolute path.

In this case, the XPath engine navigates the DOM right from the first node.

/html/body/div/div[2]/input

Question 11 : What Does A Double Slash "//" Mean In XPath?

A double (forward) Slash "//" represents the relative path.

In this case, the XPath engine searches for the matching element anywhere in the DOM.

//div//input[@id='test']

Question 12 : What Is An Absolute XPath, Explain With Example?

An absolute XPath will always search from the root node until it reaches the target. Such an XPath expression includes the single forward-slash (/) as the prefix.

/html/body/div[1]/div[5]/form/table/tbody/tr[3]/td/input

Question 13: What Is A Relative XPath, Explain With Example?

A relative XPath doesn't have a specific point to start. It can begin navigation from any node inside the DOM and continues. Such an XPath expression includes the double forward-slash (//), as given below.

//div[@id=app]

Question 14: How Do You Locate an Element By Partially Comparing Its Attributes In XPath?

XPath supports the contains () method. It allows partially matching of attribute's value.

It helps when the attributes use dynamic values while having some fixed part.

example-

xPath usage => //*[contains(@category, 'tablet')]

id=test_123

test_234

test_345

//input[contains(@id, 'test_')]

Question 15: What Is the Primary Difference Between The XPath And CSS Selectors?

With the XPath, we can traverse both forward and backward, whereas CSS selector only moves forward.

Question 16: What Does The Webdriver Driver = New FirefoxDriver(); Mean?

Webdriver driver = new FirefoxDriver();

The above line of code represents the following:

- The driver is a variable of type ‘Webdriver’ interface.
- We are instantiating an object of the FirefoxDriver class and storing it into the driver variable.

Question 17: Why Do We Create A Reference Variable of Type Webdriver, Not the Actual Browser Type?

It is because we could use the same Webdriver variable to hold the object of any browser, such as the ChromeDriver, IEDriver, or SafariDriver, etc.

We follow this approach as it can work with any browser instance.

WebDriver driver = new FirefoxDriver();

This approach is right too but will work only the Firefox.

FirefoxDriver driver = new FirefoxDriver();

Question 18:

How Do You Pass Credentials to An Authentication Popup In Selenium?

We combine the username and password strings using the colon separator and stuff them between the “http://” and the site URL. See the below example.

http://userid:passcode@gmail.com

e.g. http://userid:passcode@gmail.com

Question 19: What Are the Different Exceptions Available In Selenium?

Like other programming languages, Selenium also provides exception handling. The standard exceptions in Selenium are as follows.

- TimeoutException: This occurs if a command doesn’t finish within the specified duration.
- NoSuchElementException: This occurs if the web element with the specified attributes is not present on the page.
- ElementNotVisibleException: This occurs if the element is not visible but still there inside the DOM.
- StaleElementException: This occurs in the absence of an element that either got deleted or detached from the DOM.

Question 20: What Do You Know About an Exception Test In Selenium?

An exception test is a special exception that occurs in a test class.

Suppose we have created a test case that can throw an exception.

In this case, the @Test annotation can help us specify the exception that could occur.

Check out from the below example.

@Test(actualException = ElementNotVisibleException.class)

Question 21: How Is an Assert Different From Verify?

- Assert: It allows us to verify the result of an expression or an operation. If the "assert" fails, then it will abort the test execution and continues with the next case.
- Verify: It also operates the same as the assert does. However, if the "verify" fails, then it won't abort the test instead continues with the next step.

Question 22: What Difference Do You Make Between Soft Vs. Hard Assert in Selenium?

- Soft Assert: It aggregates the errors that occurred during the test execution. If such an assert fails, the control jumps to the next step.
- Hard Assert: It immediately responds with an AssertException and breaks the current test. After that, the next case in the sequence gets executed.

Question 23: What Are the Different Waits Available In WebDriver?

In Selenium Webdriver, the following three types of wait mechanisms are available.

Static Wait – Thread.sleep(15000);

- Implicit Wait –
- Explicit Wait –
- Fluent Wait –

Question 24: What Is Web Driver Implicit Wait?

Implicit Wait: It is a wait timeout which applies to a Webdriver instance. It implies that all actions of this instance will timeout only after waiting for a duration specified by the implicit wait.

WebDriver driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

e1

e2

e3

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

e4 e5 e6

driver.manage().timeouts().implicitlyWait(0, TimeUnit.SECONDS);

e7 e8 e9

Question 25: What Is Web Driver Explicit Wait?

Explicit Wait: It is an exclusive timeout method that works by adding code to delay the execution until a specific condition arises. It is more customizable in terms that we can set it up to wait for any suitable situation. Usually, we use a few of the pre-built Expected Conditions to wait for elements to become clickable, visible, invisible, etc.=

WebDriver driver = new ChromeDriver();

driver.get("http://gmail.com");

WebElement dynamicElement = (new WebDriverWait(driver, 15))

.until(ExpectedConditions.presenceOfElementLocated(By.id("dynamicElement")));

Question 26: How To Enter Text In The HTML Text Box Without Invoking The SendKeys()?

There is a Selenium JavascriptExecutor class that provides methods to perform actions on the HTML elements.

// Set up the JS object

JavascriptExecutor js = (JavascriptExecutor)driver;

```
// Issue command to enter the text
js.executeScript("document.getElementById('textbox').value = 'Some Text';");
```

Question 27: What Is The Method To Read The JavaScript Variable Using Selenium WebDriver?

Again, we can utilize the JavascriptExecutor class to read the value of a JS variable. See the below code.

```
// Set up the JavaScript object
JavascriptExecutor jscript = (JavascriptExecutor) webdriver;
// Read the site title
String strTitle = (String)jscript.executeScript("return document.title");
System.out.println("Webpage Title: " + strTitle);
```

Question 28: What Is The Command To Reset/Clear The HTML Text Box In Selenium Webdriver?

Selenium provides the clear() function to reset the value inside the text element.

```
WebDriver webdriver = new FirefoxDriver();
webdriver.get("https://www.google.com");
webdriver.findElement(By.xpath("<<xpath expr>>")).sendKeys("Selenium Interview Questions");
webdriver.findElement(By.xpath("<<xpath expr>>")).clear();
```

Question 29: What Is The Command To Get The Value Of A Text Box In Selenium Webdriver?

Selenium provides the getText() function to read the value inside the text element.

```
@Test
```

```
 public void readText() {
   System.setProperty("webdriver.chrome.driver", "<<chromedriver.exe>>");
   WebDriver webdriver = new ChromeDriver();
   webdriver.get("https://www.google.com");
   String strText = webdriver.findElement(By.xpath("<<xpath expr>>")).getText();
   System.out.println("Text element contains: " + strText);
 }
};
```

Question 30: What Is The Command To Get The Attribute Value In Selenium Webdriver?

Selenium provides the getAttribute(value) function to read the value inside the text element.

```
String strValue = webdriver.findElement(By.name("<<attrname>>")).getAttribute("value");
System.out.println("Attribute value is: " + strValue);
```

Question 31:

What Is the Principal Difference Between “GET” And “NAVIGATE” Methods?

Get method makes a page to load or extracts its source or parse the full text. On the contrary, the navigate method tracks the history and can perform operations like refresh, back, and forward.

For example – We like to move forward, execute some functionality and then move back to the home page.

We can achieve this by calling the Selenium’s navigate() API.

- The driver.get() method waits until the page finish loading.
- The driver.navigate() will only redirect and return immediately.

Question 32: How Do You Check for The Presence Of A Web Element After The Successful Page Load?

We can verify the presence of a web element with the following code.

While using the below function, do supply some timeout value (in seconds) to check the element in a regular interval.

```
public void checkIfElementPresent(String element, int timeout) throws Exception {
   for (int sec = 0;; sec++) {
      if (sec >= timeout)
         fail("Timeout! Couldn't locate element." + element);
      try {
         if (selenium.isElementPresent(element))
            break;
      } catch (Exception ex) {
      }
      Thread.sleep(1000);
   }
}
```

Question 33: How to Handle Multiple Popup Windows In Selenium?

Selenium provides the getWindowHandles() method, which returns the handles for all open popups.

We can store them into a <String> variable and convert it into an array.

After that, we can traverse the array and navigate to a specific window by using the below code.

```
driver.switchTo().window(ArrayIndex);
```

Alternatively, we can use the Java Iterator class to iterate through the list of handles. Find below is the code to handle multiple windows using Selenium.

```
String MyWindow = driver.getWindowHandle();
WebDriver popup = null;
Iterator<String> Windows = driver.getWindowHandles();
while(Windows.hasNext()) {
 String Window = hWindows.next();
 popup = driver.switchTo().window(hWindow);
```

```
 if (popup.getTitle().equals("HandlingMultipleWindows") {
 break;
 }
}
```

Question 34: How Can You Access A Database from Selenium?

Selenium doesn't have a direct API to access a database. Hence, we can look for its support in the programming language we choose to work with Selenium.

For illustration purposes, we used Java with Selenium.

Java provides the Connection class to initiate a connection with the database. It has a getConnection() method that we need to call. For this, we'll make a Connection Object. It'll manage the connection to the database.

Please note: An application could have one or more connections opened to a database or different databases.

Here is a short overview of the steps to be performed.

- Firstly, we establish a connection with the database.
- After that, we call the DriverManager.getConnection() method.
- This method accepts a string pointing to the database URL.
- The DriverManager class then attempts to find a driver to access the database URL.
- After it finds a suitable driver, the call to the getConnection() method succeeds.

Syntax:

```
String url = "jdbc: odbc: makeConnection";


Connection con = DriverManager.getConnection(url, "userID", "password");
```

Question 35: How To Work With AJAX Controls In WebDriver?

AJAX is an acronym for Asynchronous JavaScript and XML. It is independent of the opening and closing tags required for creating valid XML.

Sometimes, the WebDriver itself manages to work with the Ajax controls and actions. However, if it doesn't succeed, then try out the below code.

```
//Waiting for Ajax Control


WebElement AjaxCtrl = (new WebDriverWait(driver,
10)).until(ExpectedConditions.presenceOfElementLocated(By. &lt;locatorType&gt;("&lt;locator
Value&gt;")));
```

Question 36: What Is A Page Object In Selenium WebDriver?

First of all, both these terms belong to the Page Object Model (POM), a design pattern in Selenium.

Let's now see how they are different from each other.

Page Object is a class in POM corresponding to a web page. It captures the functionality as functions and objects as members.

```
public class LogInPage

{

   private WebElement user;

   private WebElement pass;


   public LogInPage() {

   }


   public void findObjects() {

      user = browser.findElement(By.id("userName"));

      pass = browser.findElement(By.id("password"));

   }


   public void processLogIn() {
```

```
        user.sendKeys("john");
        pass.sendKeys("password");
    }
}
```

Question 37: What Is A Page Factory in Selenium WebDriver?

Page Factory is a method to set up the web elements within the page object.

```
public class LogInPage
{
    @FindBy(id="userName")
    private WebElement user;

    @FindBy(id="password")
    private WebElement pass;

    public LogInPage() {
        PageFactory.initElements(browser, this); // Setup the members as browser.findElement()
    }

    public void processLogIn() {
        user.sendKeys("john");
        pass.sendKeys("password");
    }
}
```

Question 38: What Is the Right Way To Capture A Screenshot In Selenium?

Sometimes, an image than a trace log can help us identify the right reason for an error. The code in the below example will capture the image and store it in a file.

```
Import org.apache.commons.io.FileUtils;

WebDriver ins = new ChromeDriver();

ins.get("http://www.google.com/");


File screen = ((TakesScreenshot)ins).getScreenshotAs(OutputType.FILE);

// Now you can do whatever you need to do with it, for example copy somewhere


FileUtils.copyFile(screen, new File("c:\tmp\myscreen.png"));
```

Question 39: How To Resolve The SSL Certificate Issue (Secured Connection Error) In Firefox With WebDriver?

There can be many reasons for the secured connection error. It could be because of the following:

- While a site is getting developed, it may not have the right SSL certificate.
- The site may be using a self-signed certificate.
- The SSL may not have configured appropriately at the server end.

However, you still want to test the site's standard functionality using Selenium. Then, the idea is to switch off the SSL setting and ignore the SSL error.

Check out the below code to disable the SSL in Selenium.

```
FirefoxProfile ssl = new FirefoxProfile();

ssl.setAcceptUntrustedCertificates(true);

ssl.setAssumeUntrustedCertificateIssuer(false);

WebDriver ins = new FirefoxDriver(ssl);
```

Question 40: How To Handle A Proxy Using Selenium In Java?

Selenium implements a PROXY class to configure the proxy. See below example:

```
String setPROXY = "10.0.0.10:8080";

org.openqa.selenium.Proxy allowProxy = new.org.openqa.selenium.Proxy();

allowProxy.setHTTPProxy(setPROXY)

    .setFtpProxy(setPROXY)

    .setSslProxy(setPROXY);

DesiredCapabilities allowCap = new DesiredCapabilities();

allowCap.setCapability(CapabilityType.PROXY, allowProxy);

WebDriver driver = new FirefoxDriver(allowCap);
```

Question 41: What Are TestNG Annotations Frequently Used with Selenium?

TestNG annotations prioritize the calling of a test method over others. Here are the ones to use with Selenium:

- @BeforeSuite – to run before all tests.
- @AfterSuite – to run only once after all tests.
- @BeforeClass – to run only once before the first test method.
- @AfterClass – to run only once after all the test methods of the current class finish execution.
- @BeforeTest – to run before any test method inside the “Test” tag.
- @AfterTest – to run after any test method inside the “Test” tag.

Question 42:

What Do You Know About TestNG @Parameters?

In TestNG, the “@Parameters” is a keyword that allows the arguments to pass to “@Test” methods.

Please refer to this TestNG tutorial to learn more about parameters.

Question 43: What Is Meant By Grouping In TestNG?

It is an innovative TestNG feature that didn't exist in the JUnit. You can assign methods with proper context and refine groupings of test methods.

You can not only link methods to groups but also tell groups to include other groups.

Question 44: How To Associate A Single Test To Multiple Groups In TestNG?

TestNG framework allows multiple tests to run by using the test group feature.

We can associate a single test to multiple groups, as shown in the below example.

```
@Test(groups = {"regression-testing", "smoke-testing"})
```

Question 45: What Is Behavior Driven Development (BDD) Framework?

BDD follows most of the principles of TDD but replaces its unit-centric approach with a domain-centric design.

It intends to bring in inputs not only from the Dev or QA but an array of stakeholders such as Product Owners, Technical Support, Managers, and Customers.

The goal is to identify and automate appropriate tests that reflect the behavior sought by the principal stakeholders**.**

Question 46: What Are the Main Traits Of A Software Test Automation Framework?

A test automation framework should have the following features.

- Flexible in the programming language selection
- Support of keywords and actions
- Provision of data sources for input
- Allow test case creation and modification
- Define test case priority
- Manual or automated execution
- Maintain test results history
- Generate test metrics such as test vs. code coverage
- Report creation
- CI tool integration such as Jenkins

- Cross-browser and cross-platform support

Question 47: What Are Challenges Have You Faced with Selenium? And How Did You Overcome Them?

Here are some of the problems that testers usually face while doing automation with Selenium.

- Wrong implementation: I used the page object model, but had it implemented incorrectly. My classes were focusing on the web elements rather than they should have resembled the user actions.

- Duplicate code: The project had many category pages. Each category had a different search function instead of handling them at a central place.

- Ineffective use of wait: I used implicit wait with a fixed timeout. But some pages were timing out due to higher load time. I had to adopt the Fluent wait (with a variable timeout) to overcome this problem.

- Improper error handling: It was getting hard to debug the cause of a failed test. At some places, the {try-catch} blocks were missing, and hence, cases were skipping w/o giving a proper reason. Therefore, I had to refactor the code by adding asserts and exception handling.

- Inconsistent XPath: Most of the locators were using the XPath method. And the developers kept them changed while fixing new defects. I called up a discussion with them and agreed to have a fixed XPath or an ID for the web elements.

- Performance & Localization: We were using the flat files (CSV) initially to feed data to test cases. However, it had us failed in testing localization as well as beaten us on the performance. Ee migrated all of our test data to MySQL and fixed both issues.

- Monolithic tests: Earlier tests weren't using the labeling. Honestly, there wasn't a way to do it. Hence, we integrated our test suite with TestNG and got away with this limitation. Now, we have many test groups like features-based (F1, F2, F3…), priority-based (P1, P2, P3)

Question 48: What Benefits Does TestNG Have Over The JUnit Framework?

TestNG vs. JUnit – The Benefits

1 In JUnit, we start by declaring the @BeforeClass and @AfterClass. However, there isn't such a constraint in TestNG.

2 TestNG allows adding setUp/tearDown routines, which JUnit doesn't.

3 -@BeforeSuite & AfterSuite, @BeforeTest & AfterTest, @BeforeGroup & AfterGroup

4 TestNG doesn't enforce a class to extend.

5 Method names aren't mandatory in TestNG, which are in JUnit.

6 We can make a case depend on another in TestNG, whereas in JUnit, it's not possible.

7 TestNG allows grouping of tests, whereas JUnit doesn't. Executing a group will run all tests under it. For example, if we have a no. of cases distributed in two groups, namely the Sanity and Regression. If the requirement is to run the "Sanity" tests, then we can configure TestNG to execute the tests under the "Sanity" group. TestNG will execute all cases associated with the "Sanity" group.

The parallelization of test cases is another crucial feature of TestNG.

Question 49: What Type of Test Framework Did You Create Using Selenium?

While replying to such questions, stay focused, and keep your answer short and crisp. You can start by telling about the different components in your framework and then explain them one by one.

Here is an illustration for your help.

- I worked on a framework built on top of the Page Factory framework.
- I've created a page class for every web page in my application. It keeps the objects and the handler functions.
- Every page class has a follow up test class where I create tests for related use cases.
- I used separate packages to host the pages and their test classes. It's a best practice to do that way.
- The framework also had a lib package for utility and some standard wrapper functions over Selenium APIs.
- Java is ca ore programming language used for this project. It was primarily because the team had previous Java experience. Also, we could utilize the TestNG annotations and report features.
- Most test cases are data-driven. They require input from the external data source. So, I used Java property/POI class to read from the CSV/XLS files.
- We used the TestNG group feature for labeling test cases as P1, P2, and P3.
- The Log4J library provided the necessary support for tracing in our project.
- Instead of using the TestNG reporting, we preferred the Extent report. It has more graphical options and gives an in-depth analysis of the results.

- We built the framework with the help of Maven. Also, Jenkins provided support for automated build and execution.
- Bitbucket allowed us to manage our source code using git repositories.

Question 50: What Is the Principal Difference Between A Data-Driven Framework & A Keyword Driven Framework?

Data-driven framework (DDF)

- In a Data-driven framework, the test case logic is the part of test scripts.
- It advises keeping the input data separate.
- Usually, the test cases receive the data from the external files (XLS or CSV) and store them into variables. Later during execution, the variables serve as input as well as verification points.

Keyword-driven framework (KDF)

- In KDF, there happens to be a table of keywords. Every keyword either maps to an action or an object.
- The actions reflect the product functionality, and the objects represent its resources such as web elements, command or the DB URL, etc.
- They remain detached from the test automation suite. Hence, the creation of the tests can continue with or without the AUT (application under test).
- The keyword-driven tests cover the full functionality of the application under test