InterviewBit

Selenium Interview Questions

Contents

Selenium Interview Questions for Experienced

(.....Continued)

19. Explain the difference between findElement() and findElements() in Selenium.
20. Explain the pause feature in Selenium IDE.
21. With the help of code snippets, explain how we can create right-click and mouse hover actions in Selenium.
22. Can we handle a windows-based pop-up in Selenium, and if not, then what are the alternatives?
23. Can you capture a screenshot using Selenium? If yes, write a simple code to illustrate the same.
24. Explain different types of framework and connection of Selenium with Robot Framework.
25. Demonstrate usage of Selenium through a test application.
26. Explain basic steps of Selenium testing and its widely used commands via a practical application.
27. What do you understand about the Page Object Model in the context of Selenium? What are its advantages?
28. What do you understand about Jenkins? Why are the benefits of using it with Selenium?
29. How will you select a date from a datepicker in a webpage using Selenium for automated testing? Explain with a proper code.
30. What do you understand about broken links? How can you detect broken links in Selenium? Explain properly with code.
31. What do you understand about window handle in the context of automated testing? How can you handle multiple windows in Selenium?

Let's get Started

What is Selenium?

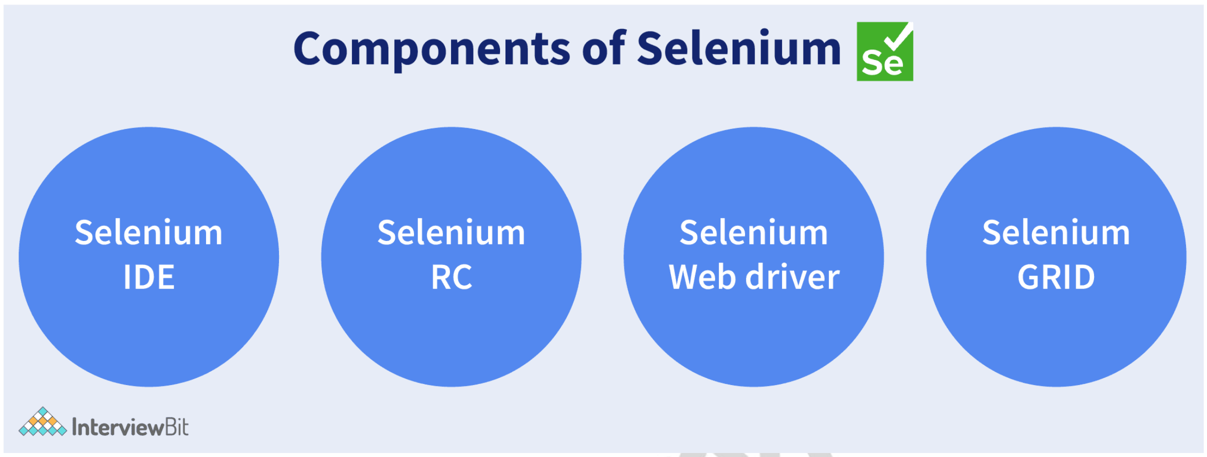
Selenium is an open-source (free) **automated testing** framework for validating web applications across multiple browsers and platforms. Selenium Test Scripts can be written in a variety of programming languages, including Java, C#, Python, and others. Selenium Testing is the term for testing done using the Selenium testing tool.

Selenium Software is a collection of tools, each of which caters to a specific organization's Selenium QA testing requirements. The following is a list of tools incorporated within Selenium:

Selenium Integrated Development Environment (IDE)

- Selenium Remote Control (RC)
- **Selenium WebDriver**
- Selenium Grid

Jason Huggins was the one to invent Selenium in 2004. He was working as an engineer at ThoughtWorks on a web application that required frequent testing. He designed a JavaScript program that would automatically control the browser's behaviour after seeing that the repetitive Manual Testing of their application was getting increasingly inefficient. This application was given the name "JavaScriptTestRunner" by him. He released JavaScriptRunner open-source, later renamed Selenium Core, after seeing promise in the notion to assist automate other online applications.



The Origin of Selenium Remote Control (Selenium RC):

Due to the limits imposed by the same-origin policy, Selenium Core testers had to install the entire application under test as well as the webserver on their own local PCs. So Paul Hammant, another ThoughtWorks engineer, decided to build a server that will function as an HTTP proxy, fooling the browser into thinking Selenium Core and the web application being tested are from the same domain. The Selenium Remote Control, or Selenium 1, was the name given to this system.

Selenium Grid's Inception:

Patrick Lightbody created Selenium Grid to answer the need to reduce test execution times as much as feasible. He dubbed the system "Hosted QA" at first. It was capable of taking browser screenshots during key stages, which can later be analysed, as well as transmitting Selenium commands to several machines at the same time.

The Origin of Selenium IDE:

Selenium IDE is a Firefox extension built by Shinya Kasatani of Japan that can automate the browser using a record-and-playback function. He came up with this concept to speed up the process of building test cases even further. In 2006, he gave the Selenium IDE to the Selenium Project.

The Origin of WebDriver:

WebDriver was designed by Simon Stewart in 2006, at a time when browsers and web applications were getting more capable while also becoming more restrictive, thanks to JavaScript tools like Selenium Core. It was the first cross-platform testing framework to allow OS-level control of the browser.

In this article, we have covered the most frequently asked interview questions on Selenium using Java as the programming language. If you are appearing for a Selenium interview, you can expect questions from automated testing and Java as well.

Selenium Interview Questions for Freshers

1. What is meant by Selenium Suite and what are its different components?

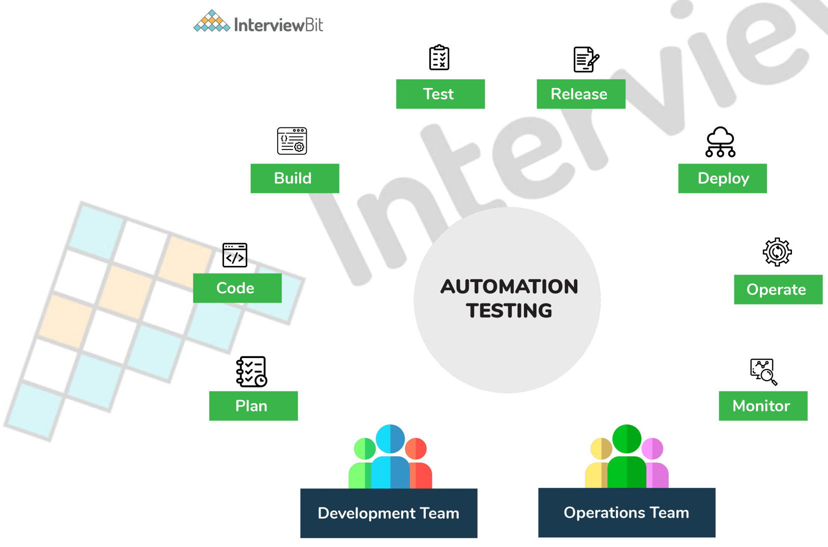
Selenium is a package of several testing tools and is therefore often referred to as a Selenium Suite with each of these tools designed to cater to a different testing requirement.

Following are the different components of Selenium Suite:

- **Selenium Integrated Development Environment (IDE)**: It is a Firefox/Chrome plug-in that is developed to speed up the creation of automation scripts by recording the user actions on the web browser and exporting them as a reusable script.
- **Selenium Remote Control (RC)**: It is a server that enables users to generate test scripts in their preferred programming language. It accepts commands from the test scripts and sends them to the browser as Selenium core JavaScript commands, for the browser to behave accordingly.
- **Selenium WebDriver:** It is a programming interface that helps create and run test cases by directly communicating with the web browser and using its native compatibility to automate. Unlike RC, it doesn’t require an additional server to create and run test cases.
- **Selenium Grid:** It allows parallel execution of tests on different browsers and operating systems by distributing commands to different machines simultaneously.

2. What is automation testing, and what are its advantages?

Automation testing or Test Automation is a process of automating the manual testing process of an application or a system by using testing tools that allow you to create scripts that can be executed repeatedly, generating detailed test reports of the application or system under test.



Advantages of Automated Testing are:

- It supports both the performance and functional testing of an application or system.
- It facilitates the execution of repeated test cases.
- It allows the parallel execution of the test cases.
- It improves the accuracy and efficiency of the system by reducing the manual intervention of humans to generate test cases.
- It helps in testing a large scale test matrix.
- It saves valuable time and money for the testing team involved in the project.

3. What are the advantages of using Selenium as an automation tool?

Following are the advantages of using Selenium for automated testing :

- **Open-Source:** Selenium's greatest strength, as previously said, is that it is a freeware and portable tool. There are no out-of-pocket expenses. The utility can be downloaded for free, and community-based help is also accessible.
- **Language assistance:** Java, Perl, Python, C#, Ruby, Groovy, JavaScript, and other languages are supported by Selenium. It has its own script, yet it is not constrained by it. It can work with a variety of languages, depending on the developers' and testers' preferences.
- **Compatible with a variety of operating systems:** Selenium may run on a variety of operating systems, including Windows, Mac OS X, Linux, and UNIX. A customized testing suite can be constructed on any platform and then executed on another using the Selenium suite of products. For example, you may write test cases in Windows and run them on a Linux system with ease.
- **Browser compatibility:** Selenium is compatible with a variety of web browsers, including Internet Explorer, Chrome, Firefox, Opera, and Safari. When running tests and testing them across multiple browsers at the same time, this becomes really useful.
- **Programming languages and framework support:** Selenium works with a variety of programming languages and frameworks. For source code compilation, it can, for example, integrate with ANT or Maven frameworks. It may also be used to test apps and generate reports using the TestNG framework. Continuous Integration (CI), can integrate with Jenkins or Hudson, and it can also integrate with other open-source tools to offer other functionalities.
- **Tests on a variety of devices:** On Android, iPhone, and Blackberry, Selenium Test Automation can be used to automate mobile web applications testing. This can aid in the generation of necessary results and the ongoing resolution of bugs present in the application.
- **Regular updates:** Selenium support is based on a community, which allows for frequent updates and upgrades. These upgrades are simple to install and don't require any special training. Selenium is thus both resourceful and cost-effective.
- **Selenium suites with a lot of content:** Selenium is more than just a single tool or utility; it's a full set of numerous testing tools that's why it's called a Suite. Each tool is tailored to specific testing needs and test environment constraints. Selenium also includes features such as Selenium IDE, Selenium Grid, and Selenium Remote Control (RC).
- **Ease with which it can be implemented:** Selenium has a user-friendly interface that makes it simple to develop and perform tests. Its open-source capabilities

4. What are the disadvantages of using Selenium as a testing tool?

The following are the disadvantages of using Selenium as a testing tool:

- **Tests web applications only**: Selenium supports the testing of only web-based applications. Mobile applications, Captcha, and Barcode readers cannot be tested using Selenium unless integrated with third-party tools like Appium and TestNG.
- **No built-in reporting and test management facility**: Selenium can generate reports only using third-party tools like TestNG or JUnit.
- **Unavailability of reliable tech support**: Since Selenium is an open-source tool, no dedicated support for user issues is available.
- **May require the knowledge of programming languages**: Some prior programming knowledge is required to use Selenium.

5. Why should Selenium be selected as a testing tool for web applications or systems?

Selenium provides the following advantages, which make it an excellent automated testing framework:

- It is free and open-source software with a large user base and supports providing community.
- It has cross-browser compatibility and supports multiple browsers like Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.
- It supports multiple operating systems such as Windows, Linux, macOS, etc.
- It facilitates the usage of multiple programming languages including Scala, Ruby, Python, PHP, Perl, Java, Groovy, C#, etc.
- It provides support for distributed testing as well.

6. Can selenium be used to launch web browsers?

Yes, Selenium provides good support to launch web browsers like Google Chrome, Mozilla Firefox, Internet Explorer, etc.

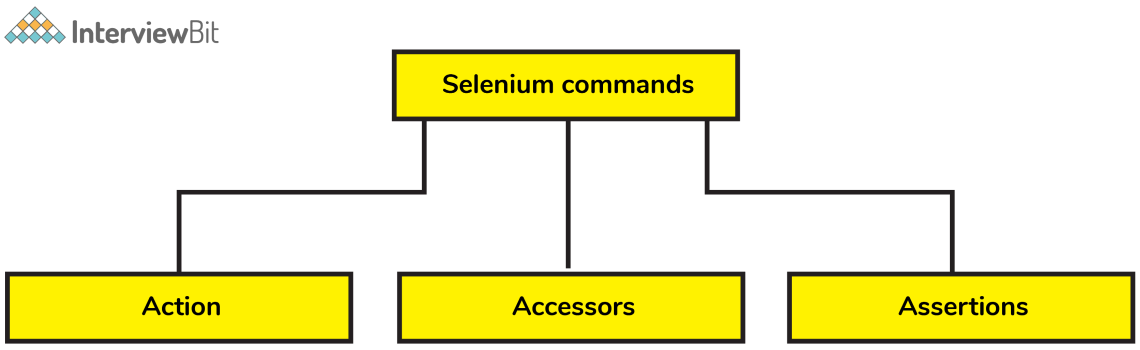
The following commands can be used to launch web browsers using Selenium:

- `WebDriver driver = new FirefoxDriver();`
- `WebDriver driver = new ChromeDriver();`
- `WebDriver driver = new InternetExplorerDriver();`

7. What is meant by Selenese? Explain different types of Selenium commands.

The language used for writing test scripts in Selenium IDE is called Selenese. It is a set of commands used to test your web application or system. Selenium commands could be divided into 3 major categories:

- **Actions:** These are the commands interacting directly with web applications.
- **Accessors:** These are the commands which allow users to store values to a user-defined variable.
- **Assertions:** They enable a comparison of the current state of the application with its expected state.



8. What is meant by a locator and name a few different types of locators present in Selenium.

A locator is an address for uniquely identifying web elements within a web page. There are different types of locators present in Selenium to identify web elements uniquely and accurately like:

- ID
- ClassName
- Name
- TagName
- LinkText
- PartialLinkText
- Xpath
- CSS Selector
- DOM.

9. State the major difference between “assert” and “verify” commands in Selenium.

Both “assert” and “verify” commands check whether the given condition is true or false and the only difference between them is that:

- **Assert:** Assert condition stops the execution of the testing if the given condition is false else would continue with the further tests.
- **Verify:** Verify the condition doesn’t stop the flow of execution irrespective of the condition being true or false.

10. What is meant by an exception test in Selenium?

An exception test is a test that expects an exception to be thrown inside a test class. It expects a @Test annotation followed by the expected exception name in the brackets.

Eg: `@Test(expectedException = NoSuchElementException.class)` is an exception test for missing elements in Selenium.

11. What is meant by XPath in Selenium. Explain XPath Absolute and XPath Relative.

XPath, also defined as XML-Path (Extensible Markup Language Path), is a language used to query XML documents and provide functionalities like locating elements in Selenium by iterating through each element in a webpage. In XPath, data is stored in a key-value pair format similar to an HTML tag. It uses a single slash, i.e. ‘ / ’ for creating an absolute path, and a double slash, i.e. ‘ // ’ for creating a relative path for an element to be located on a webpage.

12. In Xpath, what is the difference between "/" and "//"?

Single Slash "/" - A single slash is used to create an Xpath with an absolute path, i.e. the XPath will begin with the document node/start node.

For example,

```
Absolute XPath: /html/body/div/div/form/input
```

Here, /html is the root html node.

Double Slash "//" - The double slash is used to construct an Xpath with a relative path, which means the XPath can start selection from anywhere on the page.

For example,

```
Relative XPath: //input[@id = 'email']
```

Here, we can locate an input having id = ‘email’ present anywhere in the document object model (DOM).

13. What is the difference between the commands "type" and "typeAndWait" in the context of Selenium?

The "type" command is used to enter keyboard key values into a software web application's text box. It can also be used to choose values from a combo box, whereas the "typeAndWait" command is used when you finish typing and the software web page begins to reload. This command will wait for the page of the software program to reload before proceeding. You must use a basic "type" command if there is no page reload event when typing.

14. Differentiate between findElement() and findElements() in the context of Selenium with proper examples.

Following table lists the differences between findElement() and findElements() in Selenium :

findElement()	findElements()
The first web element that matches the locator is returned.	It gives you a list of all the web items that match the locator.
If there are no matching web elements, a NoSuchElementException is produced.	If there are no matching elements, an empty list is returned.
Syntax – `WebElement button = webdriver.findElement(By.name("<<Name value>>"));`	Syntax – `List<WebElement> buttons = webdriver.findElements(By.name("<<Name value>>"));`

- Using `findElements()` :-

```
// JAVA
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.concurrent.TimeUnit;
public class findElements {
  public static void main(String[] args) {
     System.setProperty("webdriver.chrome.driver", "C:\\Users\\vaibhav\\Desktop\\Java\\
     WebDriver driver = new ChromeDriver();
     String url = "https://www.exampleurl.com/example.htm";
     driver.get(url);
     driver.manage().window().maximize();
     driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

     List<WebElement> rows = driver.findElements(By.xpath("//table/tbody/tr[2]/td")); /
     System.out.println("The number of values in row 2 is "+ rows.size());
     driver.close();
  }
}
```

Explanation - In the above code, first of all, we import all the necessary headers and then set up the driver for the chrome browser. We use the `findElements()` method to find all the values present in the 2nd row of a table in the given URL web page using the XPath of the element.

- Using `findElement()` :-

```
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.concurrent.TimeUnit;
public class findTagname {
  public static void main(String[] args) {
     System.setProperty("webdriver.chrome.driver",    "C:\\Users\\vaibhav\\Desktop\\Jav
     WebDriver driver = new ChromeDriver();
     String url = "https://www.exampleurl.com/example.htm";
     driver.get(url);
     driver.manage().timeouts().implicitlyWait(12, TimeUnit.SECONDS);
     driver.findElement(By.cssSelector("input[id='search']")).sendKeys("Selenium"); //U
     driver.close();
  }
}
```

Explanation - In the above code, first of all, we import all the necessary headers and then set up the driver for the chrome browser. We use the findElement() method to find an input element having an id attribute set as search.

15. In Selenium, how will you wait until a web page has been loaded completely?

There are two methods of making sure that the web page has been loaded completely in Selenium.

They are as follows :

1. Immediately after creating the webdriver instance, set an implicit wait:

```
temp = driver.Manage().Timeouts().ImplicitWait;
```

On every page navigation or reload, this will try to wait until the page is fully loaded.

2. Call JavaScript return document.readyState till "complete" is returned after page navigation. As a JavaScript executor, the web driver instance can be used.

Code example:

```
new WebDriverWait(firefoxDriver, pageLoadTimeout).until(
     webDriver -> ((JavascriptExecutor) webDriver).executeScript("return document.ready
```

Selenium Interview Questions for Experienced

16. Explain the difference between driver.close() and driver.quit() command in Selenium?

Following is the major difference between both the commands:

- `driver.close()` command closes the currently active window on which the user is working or the window being currently accessed by the web driver.
- `driver.quit()` command, unlike the `driver.close()` command it closes all the windows opened by the program and hence should be used with care.

Both the commands don't take any parameter and don't return any value either.

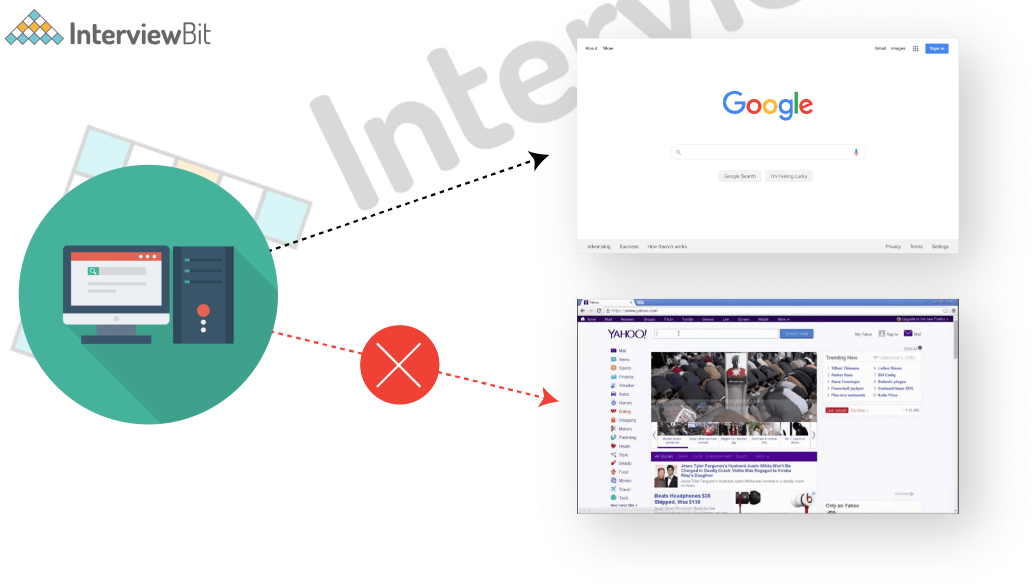
17. Explain the various navigation commands supported by Selenium?

Selenium has the support of majorly 4 navigation commands:

- `navigate().back()` : This command is used for taking the user to the last webpage of the browser history.
- `navigate().forward()` : This command is used for taking the user to the next web page of the browser history.
- `navigate().refresh()` : This command is used for reloading the web components of a webpage by refreshing it.
- `navigate().to()` : This command is used for navigating to a particular URL in a new web browser. It takes the URL to be migrated to, as a parameter.

18. Explain the same-origin policy and how Selenium handles it?

Same Origin policy is a feature adopted for security purposes that allows a web browser to run scripts from one webpage to access the contents of another webpage provided both the pages have the same origin. The URL scheme, hostname, and port number combo are referred to as the origin. This policy was introduced to prevent access to sensitive data on one webpage by another for ill purposes. Consider a Java program used by scaler.com, the program can access domain pages like scaler.com/mentors, scaler.com/courses but none from different domains like facebook.com.



The Selenium Server (Selenium RC) acts as a client configured HTTP proxy and "tricks" the browser into believing that Selenium Core and the web application being tested come from the same origin.

19. Explain the difference between findElement() and findElements() in Selenium.

Following is the major difference between the **two commands**:

1. `findElement()` : command is used for finding a particular element on a web page, it is used to return an object of the first found element by the locator. Eg:

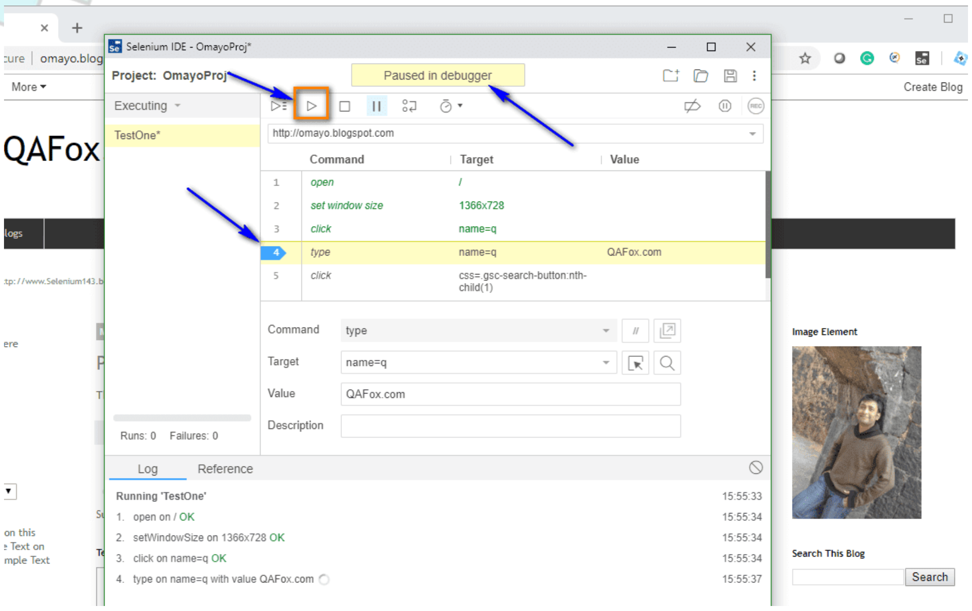
```
WebElement element = driver.findElement(By.id(example));
```

2. `findElements()` : command is used for finding all the elements in a web page specified by the locator value. The return type of this command is the list of all the matching web elements. Eg:

```
List <WebElement> elementList = driver.findElements(By.id(example));
```

20. Explain the pause feature in Selenium IDE.

The pause feature is built to handle exceptions in the test script by allowing the user to pause at the statement causing the exception and enter the debug mode by clicking on the pause icon on the top right corner of the IDE. This feature prevents the entire test case's failure and gives the user a chance to correct the error instantly.



	Command	Target	Value
1	open	/	
2	set window size	1366x728	
3	click	name=q	
4	type	name=q	QAFox.com
5	click	css=.gsc-search-button:nth-child(1)	

Command: type
Target: name=q
Value: QAFox.com
Description:

Runs: 0 Failures: 0

Log Reference

```
Running 'TestOne'                                15:55:33
1. open on / OK                                  15:55:34
2. setWindowSize on 1366x728 OK                  15:55:34
3. click on name=q OK                            15:55:34
4. type on name=q with value QAFox.com           15:55:37
```

21. With the help of code snippets, explain how we can create right-click and mouse hover actions in Selenium.

The following code can replicate right-click action:

```
actions action = newActions(driver);
WebElement element = driver.findElement(By.id("elementId"));
action.contextClick(element).perform();
```

The following code can replicate mouse hover action:

```
actions action = newActions(driver);
WebElement element = driver.findElement(By.id("elementId"));
action.moveToElement(element).perform();
```

22. Can we handle a windows-based pop-up in Selenium, and if not, then what are the alternatives?

No, Selenium doesn't support windows based pop-ups as it's an automated testing tool built for web application based testing. However, with the support of third-party tools like AutoIT, Robot class, etc., windows-based pop-ups can be handled in selenium.

23. Can you capture a screenshot using Selenium? If yes, write a simple code to illustrate the same.

Yes, using a web driver in Selenium, we can capture the screenshot. Following is the code to do the same:

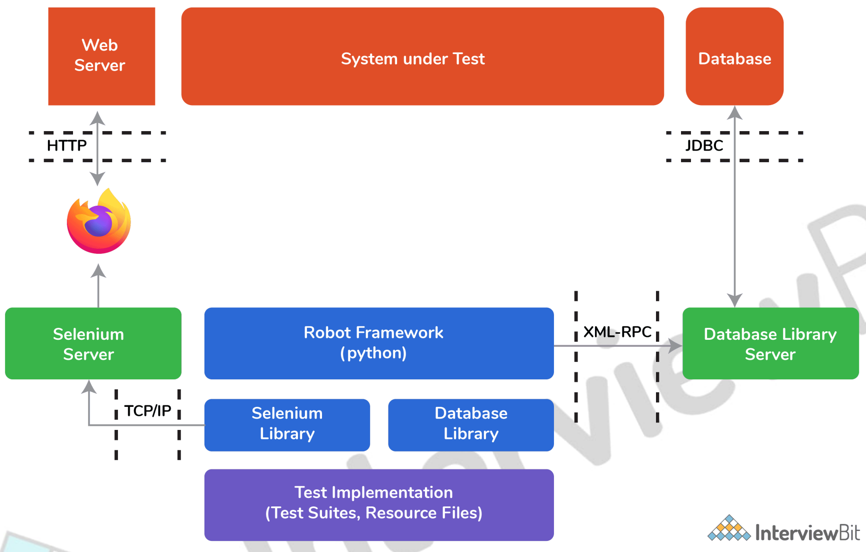
```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import java.io.File;
import java.io.IOException;
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class TakeScreenshot {
WebDriver drv;
   @ Before
   public void setUp() throws Exception {
   driver = new FirefoxDriver();
   drv.get("https://google.com");
}
   @ After
   public void tearDown() throws Exception {
   drv.quit();
   }
   @ Test
   public void test() throws IOException {
   // Capture the screenshot
   File scrFile = ((TakeScreenshot)drv).getScreenshotAs(OutputType.FILE);
   // Code for pasting screenshot to a user-specified location
   FileUtils.copyFile(scrFile, new File("C:\\Screenshot\\Scr.jpg"))
   }
}
```

24. Explain different types of framework and connection of Selenium with Robot Framework.

Following are the different types of frameworks:

- **Behavior-Driven Development Framework:** This type of framework provides a readable and easily understandable format to Business Analysts, Developers, Testers, etc.
- **Data-Driven Testing Framework:** This type of framework helps separate test data from the test-script logic by storing test data in some external database in the form of key-value pairs. These keys can be used for accessing as well as populating data into the test scripts.
- **Keyword-Driven Testing Framework:** This type of framework is an extension of the data-driven testing framework. In addition to separating test data and the test-script logic, it also separates a part of the test script code by storing it in an external data file.
- **Library Architecture Testing Framework:** This type of framework groups common steps into functions under a library and calls these functions as and when required.
- **Module-Based Testing Framework:** This type of framework divides each test application into several isolated and logical modules, with each module having its distinct test script.
- **Hybrid Testing Framework:** This type of framework is a combination of the above-mentioned frameworks leveraging all their good features.
- Robot Framework is a modular open-source automation framework that can interact with 3rd party libraries and functions. To execute a web testing library such as Selenium, a test automation runner or an automation wrapper is required, which is provided to it in the form of Robot Framework. Other popular test runners to serve the same purpose are MSTest, TestNG, Nunit, Junit, etc.

The below diagram shows the connection of the Robot framework to the Selenium library:



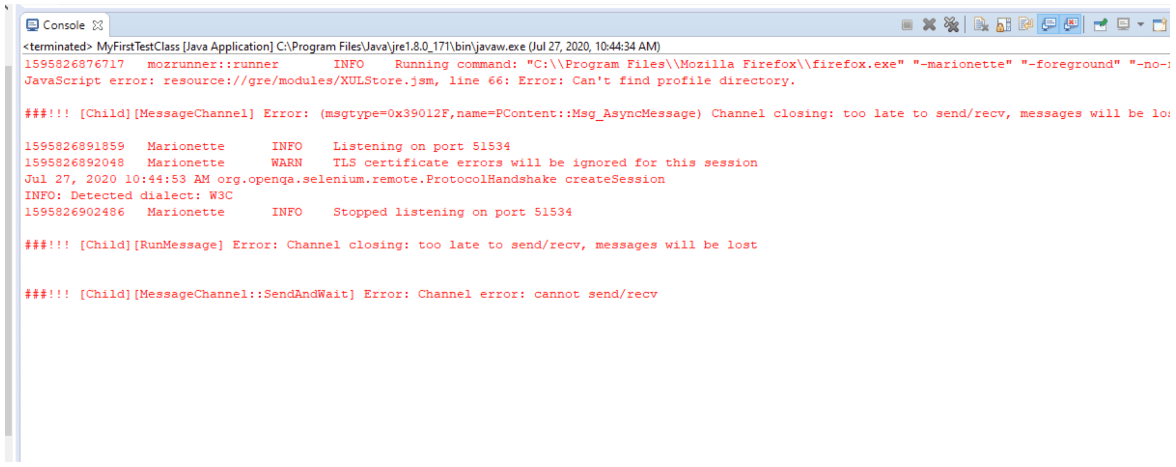
25. Demonstrate usage of Selenium through a test application.

You need the following prerequisites to run a demo Selenium test script:

- **Java SDK** in your respective Operating System.
- A **Java-based IDE** such as Eclipse or IntelliJ.
- A **Selenium WebDriver** to be added as a dependency to Java IDE.

```
package scalerAcademy;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.WebDriver;
public class MyFirstTestClass {
public static void main(String[] args) throws InterruptedException {
//It sets the system property to the given value.
System.setProperty("webdriver.gecko.driver","D:\\Softwares\\geckodriver.exe”);
WebDriver driver = new FirefoxDriver();
         driver.get("https://www.google.com/");
         //Launch website in the browser
         driver.manage().window().maximize();
 //The sleep pauses the execution of the thread for 5000 ms.
         Thread.sleep(5000);
         driver.quit();
 }
}
```

Once you run the above script in a Java IDE, you'll get the following execution logs displayed in your IDE window.



```
<terminated> MyFirstTestClass [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Jul 27, 2020, 10:44:34 AM)
1595826876717   mozrunner::runner       INFO    Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "-marionette" "-foreground" "-no-
JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile directory.

###!!! [Child][MessageChannel] Error: (msgtype=0x39012F,name=PContent::Msg_AsyncMessage) Channel closing: too late to send/recv, messages will be lo

1595826891859   Marionette      INFO    Listening on port 51534
1595826892048   Marionette      WARN    TLS certificate errors will be ignored for this session
Jul 27, 2020 10:44:53 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
1595826902486   Marionette      INFO    Stopped listening on port 51534

###!!! [Child][RunMessage] Error: Channel closing: too late to send/recv, messages will be lost


###!!! [Child][MessageChannel::SendAndWait] Error: Channel error: cannot send/recv
```

26. Explain basic steps of Selenium testing and its widely used commands via a practical application.

Selenium testing can be divided into the following seven basic elements:

1. Creating an instance of a Webdriver: This is the first step for all the usages of a Selenium webdriver API. An instance of a webdriver interface is created using a constructor of a particular browser. This webdriver instance is used to invoke methods and to access other interfaces. Following are the most commonly used commands for initialising a web driver:

```
Firefox:
WebDriver driver = new FirefoxDriver();
Chrome:
WebDriver driver = new ChromeDriver();
Safari Driver:
WebDriver driver = new SafariDriver();
Internet Explorer:
WebDriver driver = new InternetExplorerDriver();
```

2. Navigating to a webpage: The second step after initializing an instance of a webdriver, to navigate to a particular webpage you want to test. Following are the most commonly used commands for webpage navigation:

```
Navigate to URL:
driver.get(“https://www.interviewbit.com”)
driver.navigateo.to(“https://www.interviewbit.com”)
Refresh page:
driver.navigate().refresh()
Navigate forward in browser history:
driver.navigate().forward()
Navigate backward in browser history:
driver.navigate().backward()
```

3. Locating an HTML element on the webpage: To interact with a web element and perform actions on it like clicking a button or entering text, we first need to locate the desired elements such as the button or the textbox on the web page. Following are the most commonly used commands for web element navigation:

```
Locating by ID:
driver.findElement(By.id("q")).sendKeys("Selenium 3");
Location by Name:
driver.findElement(By.name("q")).sendKeys ("Selenium 3");
Location by Xpath:
driver.findElement(By.xpath("//input[@id==’q’])).sendKeys("Selenium 3");
Locating Hyperlinks by Link Text:
driver.FindElement(By.LinkText("edit this page")).Click();
Locating by ClassName
driver.findElement(By.className("profileheader"));
Locating by TagName
driver.findElement(By.tagName("select')).click();
Locating by LinkText
driver.findElement(By.linkText("NextPage")).click();
Locating by PartialLinkText
driverlindElement(By.partialLinkText(" NextP")).click();
```

4. Performing actions on an HTML element: Once we have located the HTML element, the next step is interacting with it. Following are the most commonly used commands for performing actions on HTML element:

```
Entering a username
usernameElement.sendKeys("InterviewBit");
Entering a password
passwordElement.sendKeys("Raw");
Submitting a text input element
passwordElement.submit();
Submitting a form element:
formElement.submit();
```

5. Anticipating browser response from the action: Once an action is performed, anticipating a response from the browser to test comes under this step. It takes a second or two for the action to reach the browser, and hence wait is often required for this step. There are two main types of wait conditions:

Implicit Wait: It sets a fixed, definite time for all the webdriver interactions. It's slightly unreliable as web driver response times are usually unpredictable. Eg:

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

Explicit Wait: This type of wait condition sets an expected condition to occur on the web page or a maximum wait time for all the webdriver interactions. Eg:

```
WebElement messageElement = wait.until(      ExpectedConditions.presenceOfElementLocate
```

6. Running tests and recording their results using a test framework: in this step, we run tests in an automated test script to evaluate an application's function and performance. Various test frameworks are used for this step, such as:

1. JUnit for Java
2. NUnit for C#
3. Unittest or Pyunit for Python
4. RUnit for Ruby

Most frameworks use some sort of asset statement to verify their test results from the expected results. Eg:

```
assertEquals (expectedMessage, actualMessage);
```

7. Concluding a test: In this step, we conclude a test by invoking a quit method on the driver variable. This step closes all the webpages, quits the WebDriver server, and releases the driver. Eg:

```
driver.quit();
```

The following is an example of an app that covers all the steps mentioned above:

```
import org.openqa.selenium.By,
import org.openqa.selenium.WebElement,
import org.openqa.selenium.support.ni.ExpectedConditiof, import org.openqa.selenium.sup
import org.junit.Assert;
public class Example {
public static void main(String[] args) {
// Creating a driver instance
WebDriver driver = new FirefoxDriver(),
// Navigate to a web page
driver.get("http://www.foo.com");
// Enter text to submit the form
WebElement usernameElement = driver.findElement( By.name("username"));
WebElement passwordElement = driver.findElement(By.name(”password"));
WebElement formElement = driver.findElement(By.id(”loginForm"));
usernameElement.sendKeys("Scaler Academy");
passwordElement.sendKeys("Raw");
formElement.submit();      // submit by form element

//Putting an explicit wait
WebDriverWait wait = new WebDriverWait(driver, 10);
WebElement messageElement = wait.until(
      ExpectedConditions.presenceofElementLocated(By.id(”loginResponse"))
      ) ;
// Run a test
String message              = messageElement.getrept();
String successMsg      = "Welcome to foo. You logged in successfully.”;
Assert.assertEquals (message, successMsg);
// Conclude a test
driver.quit();
}
}
```

27. What do you understand about the Page Object Model in the context of Selenium? What are its advantages?

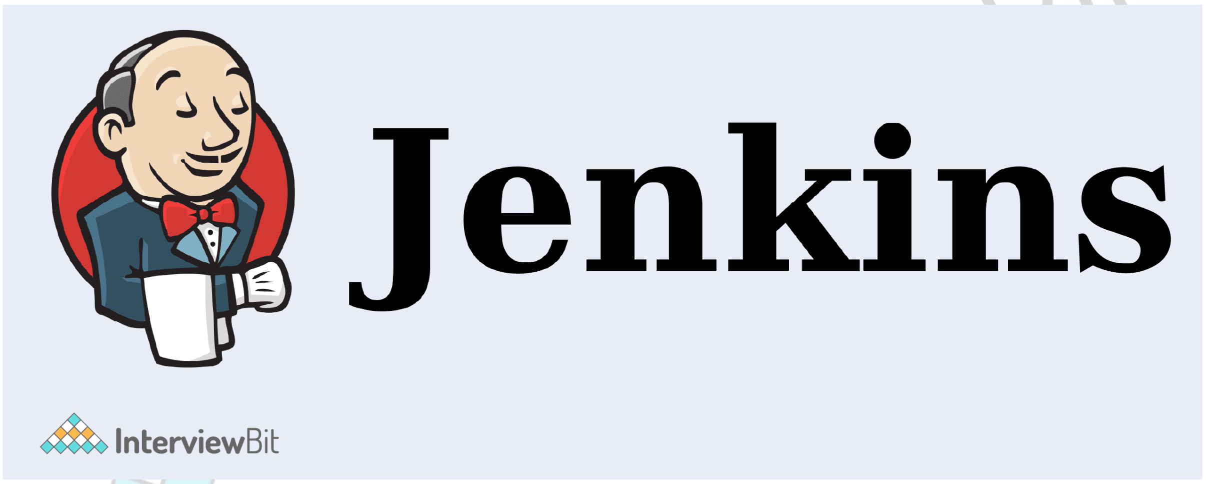
Page Object Model (POM) is a design pattern that generates an Object Repository for web UI elements and is widely used in test automation. The paradigm has the advantage of reducing code duplication and improving test maintenance. According to this paradigm, each web page in the application should have its own Page Class. This Page class will identify the web page's WebElements and also has Page methods that operate on those WebElements. The names of these methods should correspond to the tasks they perform, for example, if a loader is waiting for the payment gateway to appear, the POM method name could be `waitForPaymentScreenDisplay()` .

Following are the advantages of the Page Object Model (POM) :

- According to the Page Object Design Pattern, user interface activities and flows should be separated from verification. Our code is clearer and easier to understand as a result of this notion.
- The second advantage is that the object repository is independent of test cases, allowing us to reuse the same object repository with different tools. For example, we can use Selenium to combine Page Object Model with TestNG/JUnit for functional testing and JBehave/Cucumber for acceptability testing.
- Because of the reusable page methods in the POM classes, code gets less and more efficient.
- Methods are given more realistic names that can be easily associated with the UI operation. If we land on the home page after clicking the button, the function name will be 'gotoHomePage()'.

28. What do you understand about Jenkins? Why are the benefits of using it with Selenium?

Hudson Lab's Jenkins is the most popular open-source continuous integration technology. It's cross-platform, meaning it can run on Windows, Linux, Mac OS, and Solaris. Jenkins is a Java application. Jenkin's main purpose is to keep track of any job, such as SVN (Apache Subversion) checkouts, cron jobs, or application states. When a specific event in an application occurs, it triggers pre-configured actions.



Following are the **features of Jenkins**:

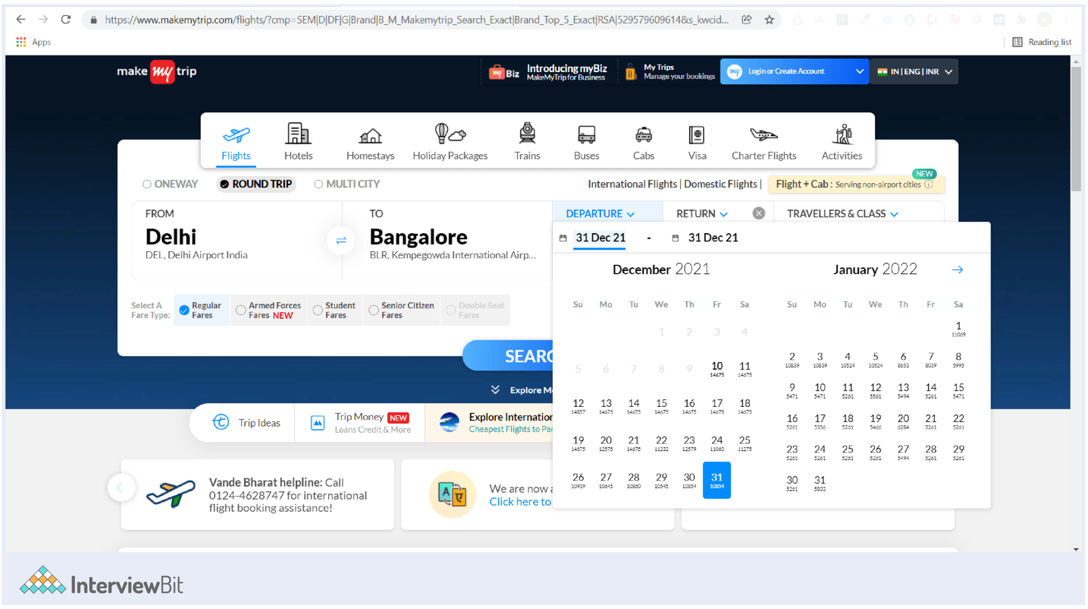
- Jenkins generates a list of all changes made in SVN repositories, for example.
- Jenkins gives permanent links to the most recent build or failed build, which can be utilised for convenient communication.
- Jenkins is simple to install using either a direct installation file (exe) or a war file for deployment via the application server.
- Jenkins can be set up to send the content of the build status to an email address.
- Simple Configuration: Jenkins makes it simple to set up multiple tasks.
- Jenkins can be configured to run the automated test build on TestNg following every SVN build.
- Jenkins documents the details of the jar, its version, and the mapping of build and jar numbers.
- Plugins: Jenkins can be set to utilise features and additional functionality provided by third-party plugins.

Following are the reasons we **use Jenkins with Selenium** :

- When you run Selenium tests in Jenkins, you can run them every time your program changes, and when the tests pass, you may deploy the software to a new environment.
- Jenkins may execute your tests at a predetermined time.
- The execution history as well as the Test Reports can be saved.
- Jenkins allows you to develop and test a project in continuous integration using Maven.

29. How will you select a date from a datepicker in a webpage using Selenium for automated testing? Explain with a proper code.

In such types of questions, the interviewer wants to assess how clear your understanding is about the framework. It is a good practice to explain the code while you write it so that the interviewer is engaged at all points and does not feel left out. We will be considering an example on MakeMyTrip.



Here, we will be using the chrome browser and so we will be implementing the code for the chrome browser only. You can implement similar code for firefox and other browsers as well.

First of all, we create a package named browserSelection which contains a class defined for handling different types of browsers such as chrome, firefox that we may want to use.

```
package browserSelection;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class SelectBrowser
{
  static WebDriver driver;
  public static WebDriver useChrome()
  {
     System.setProperty("webdriver.chrome.driver", "E:\\SeleniumLibs\\\\chromedriver_wi
     driver = new ChromeDriver();
     driver.manage().window().maximize();
     return driver;
  }
}
```

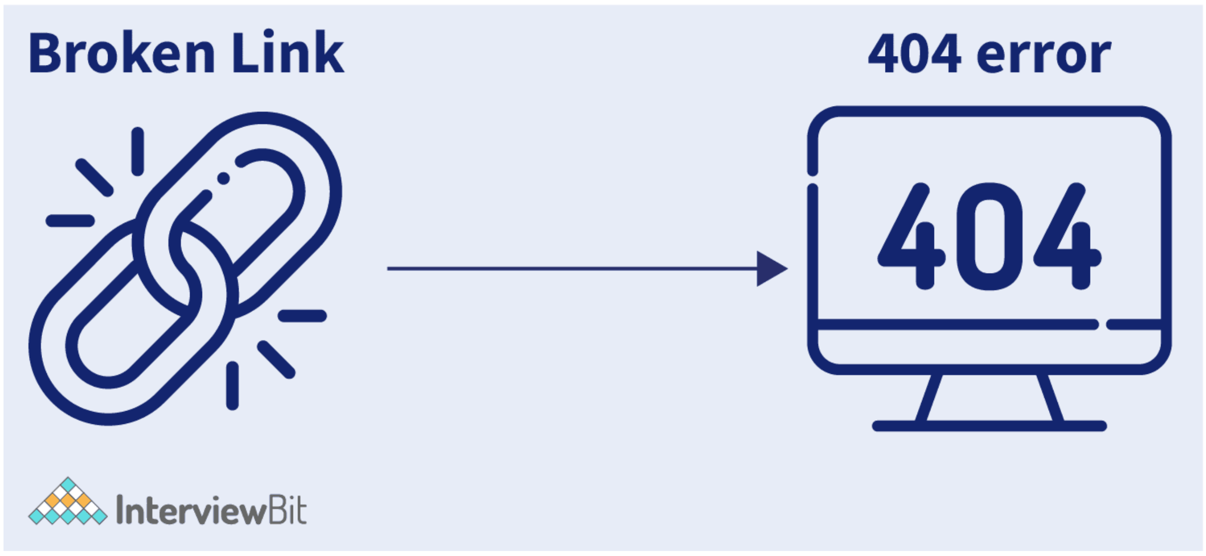
Next, we create a package named datepicker which will contain a class containing methods defined for selecting a specific date on the website of MakeMyTrip. We need to import this package into our driver class and call the respective methods.

```
package datepicker;
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebDriverException;
import org.openqa.selenium.WebElement;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
import browserSelection.SelectBrowser;
public class DatePick
{
  WebDriver driver;
  @BeforeMethod
  public void startBrowser()
  {
     driver = SelectBrowser.useChrome();
  }
  @Test
  public void selectDateUtil() throws InterruptedException, AWTException
  {
     //Modify Wait time as per the Network Ability in the Thread Sleep method
     driver.get("https://www.makemytrip.com/");
     driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
     Thread.sleep(5000);
     try
     {
        driver.findElement(By.xpath("//input[@id='hp-widget__depart']")).click();
        Thread.sleep(2000);
        Date sampleDate = new Date(); // initialising the date object with the current
        SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM yyyy");
        String date = formatter.format(sampleDate); // formatting the date object in dd
        String splitter[] = date.split("-");
        String monthYear = splitter[1]; // storing the month and year concatenated stri
        String day = splitter[0]; // storing the day number in the current date
        System.out.println(monthYear);
        System.out.println(day);

        selectDate(monthYear,day); // function invocation
        Thread.sleep(3000);

        public void selectDate(String monthYear, String select_day) throws InterruptedE
        {
           List<WebElement> elements = driver.findElements(By.xpath("//div[@class='ui-d
           for (int i=0; i<elements.size();i++)
           {
              System.out.println(elements.get(i).getText());
              //Selecting the month
              if(elements.get(i).getText().equals(monthYear))
              {
                 //Selecting the date
                 List<WebElement> days = driver.findElements(By.xpath("//div[@class='ui
```

In the above code, the function `startBrowser()` is used to invoke the `useChrome()` method from the imported package browserSelection. The function `selectDateUtil()` is used to select the current date from the date picker of the sample web page. The `endBrowser()` function is used to close the driver connections by invoking the `quit()` method.

30. What do you understand about broken links? How can you detect broken links in Selenium? Explain properly with code.

Links or URLs that are not reachable are known as broken links. They may be unavailable or inoperable due to a server issue. A URL's status will always be 2xx, indicating that it is legitimate. There are a variety of HTTP status codes, each with its own set of functions. HTTP status 4xx and 5xx indicate an invalid request. The 4xx class of status codes is used for client-side errors, while the 5xx class is used for server response errors.



You should always check for broken links on your site to ensure that the user does not end up on an error page. If the rules aren't updated appropriately, or the requested resources aren't available on the server, the error will occur. Manual link checking is a time-consuming task because each web page may have a huge number of links, and the process must be performed for each page.

To find broken links in Selenium, follow the instructions below.

- Using the <a> (anchor) tag, collect all of the links on a web page.
- For each link, send an HTTP request.
- Make that the HTTP response code is correct.
- Based on the HTTP response code, determine whether the link is genuine or not.
- Repeat the procedure for all of the links that were captured in the first step.

```
package SeleniumPackage;
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Iterator;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
public class BrokenLinks {

   public static void main(String[] args) {

       String pageURL = "http://www.interviewbit.com";
       String url = "";
       HttpURLConnection huc = null;
       int responseCode = 200;
       System.setProperty("webdriver.chrome.driver", "C:\\Users\\user\\Downloads\\seler
       ChromeOptions options = new ChromeOptions();
       options.addArguments("--headless", "--disable-gpu", "--window-size=1920,1200","-
       WebDriver driver = new ChromeDriver(options);//Creating an instance of the WebDr

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

       driver.get(pageURL);

       List<WebElement> links = driver.findElements(By.tagName("a")); // getting hold c

       Iterator<WebElement> it = links.iterator();
       // Iterating over the obtained list of elements and checking them one by one
       while(it.hasNext()){

           url = it.next().getAttribute("href");

           System.out.println(url);

           if(url == null || url.isEmpty()){
               System.out.println("The linked element has invalid href url.");
               continue;
           }

           if(!url.startsWith(pageURL)){
               System.out.println("URL belongs to another domain, skipping it.");
               continue;
           }

           try {
               huc = (HttpURLConnection)(new URL(url).openConnection());

               huc.setRequestMethod("HEAD");

               huc.connect(); // connecting to the url

               responseCode = huc.getResponseCode(); // reading the response code on fi
```

Explanation - In the above code, we first set up the system properties and then initialise a webdriver object. We find all the elements in the web page having the anchor tag with the help of the `findElements()` method. Then, we iterate over the list obtained one by one and fire up the URL and read the response code received to check if it is a broken link or not.

31. What do you understand about window handle in the context of automated testing? How can you handle multiple windows in Selenium?

Window handle is a one-of-a-kind identifier that contains the addresses of all of the windows. Consider it a window pointer that returns the string value. Each browser will presumably have its own window handle. This window handle function aids in the retrieval of all window handles.
Syntax :

- `get.windowhandle()` : This function is used to retrieve the current window's handle.
- `get.windowhandles()` : This function is useful for retrieving the handles of all the windows that have been opened.
- set: This method allows you to set the window handles as a string.
  `set<string> set= driver.get.windowhandles()`
- switch to: This method aids in the switching of windows.
- action: This method aids in the execution of specific window actions.

Let us consider an example code to understand better. We will open the website of InterviewBit and then click on all the links available on the web page. Then, we will switch from the parent window to multiple different child windows and then switch back to the parent window at last.

```
package SeleniumPackage;
import java.util.Iterator;
import java.util.Set;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class WindowHandle_Demo {
  public static void main(String[] args) throws Exception {

     System.setProperty("webdriver.chrome.driver", "C:\\Users\\user\\Downloads\\seleniu
     WebDriver driver = new ChromeDriver();
     driver.manage().window().maximize();
     // Loading the website
     driver.get("http://www.interviewbit.com/");
     String parent=driver.getWindowHandle(); // storing the parent window name as a str
     List<WebElement> links = driver.findElements(By.tagName("a")); // storing the list
     Iterator<WebElement> it = links.iterator();
     // Iterating over the list elements one by one and clicking all the links to open
     while(it.hasNext()){
        it.next().click();
     }
     Set<String> s = driver.getWindowHandles(); // Storing the list of all the child wi
     Iterator<String> I1= s.iterator();
     // Iterating over the list of child windows
     while(I1.hasNext())
     {
        String child_window=I1.next();
        if(!parent.equals(child_window))
        {
           driver.switchTo().window(child_window);
           System.out.println(driver.switchTo().window(child_window).getTitle());
           driver.close();
        }
     }
     //switch to the parent window
     driver.switchTo().window(parent);
  }
}
```

In the above code, we open the landing page of interviewbit and then find all the elements having the anchor tag and click them to open multiple child windows. Then, we iterate over each of the child windows and print them as a string. Finally, having traversed over the entire list, we break from the loop and switch back to the parent window.

Links to More Interview Questions