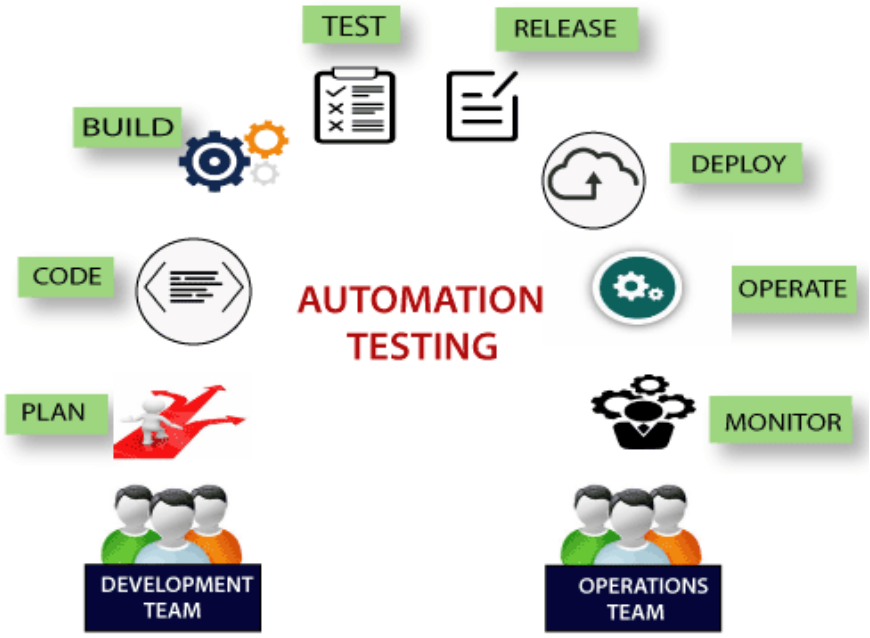
1) What is test automation or automation testing?

Test automation is the process of using automation tools to maintain test data, execute tests, and analyze test results to improve software quality. Automated testing is also called test automation or automated QA testing. When executed well, it relieves much of the manual requirements of the testing lifecycle.

Automation testing uses automation tools to write and execute test cases, no manual involvement is necessary for executing an automated test suite. Testers prefer automation tools to write test scripts and test cases and then group into test suites.

Automation testing enables the use of specialized tools to automate the execution of manually designed test cases without any human intervention. Automation testing tools can access the test data, controls the execution of tests and compares the actual result against the expected result. Consequently, generating detailed test reports of the system under test.



2) What are the advantages of automation testing?

Benefits of Automation Testing

Automation testing supports both functional and performance test on an application.

It supports the execution of repeated test cases.

It facilitates parallel execution.

It aids in testing a large test matrix.

It improves accuracy because there are no chances of human errors.

It saves time and money.

Saving Costs.

Faster Feedback Loop.

Better Allocation of Resources.

Guarantees Higher Accuracy.

Increased Test Coverage.

Detects bugs earlier.

Test at Scale.

Maximizes ROI.

3) Name some of the commonly used Automation Testing tools that are used for Functional Automation.

In functional automation Each functionality of a software application is tested by providing a different set of data inputs verifying the resulting output is as expected by comparing the actual results with the expected results. Automated functional testing is a technique that executes test cases automatically using some specific code.

Lists of top 10 used automation testing tools for Functional Automation are as follows.

Teleric Test Studio, Developed by Teleric.

TestingWhiz

HPE Unified Functional Testing (HP - UFT formerly QTP)

Tosca Testsuite

Watir

Quick Test Professional, provided by HP.

Rational Robot, provided by IBM.

Coded UI, provided by Microsoft.

Selenium, open source.

Auto It, Open Source.

4) Name some of the commonly used Automation Testing tools that are used for Non-Functional Automation.

Non functional testing is a type of software testing that verifies non functional aspects of the product, such as performance, stability, and usability. Whereas functional testing verifies whether or not the product does what it is supposed to, non functional testing verifies how well the product performs

Tools such as Apache JMeter, Gatling, and LoadRunner, help simulate high user loads and measure system performance under heavy traffi

5) What is Selenium?

Selenium is an open-source, automated testing tool used to test web applications across various browsers. Selenium can only test web applications, unfortunately, so desktop and mobile apps can't be tested. However, other tools like Appium and HP's QTP can be used to test software and mobile applications

6) What are the different components of Selenium?

Selenium is not just a single tool but a suite of software's, each having a different approach to support automation testing. It comprises of four major components which include:

Selenium Integrated Development Environment (IDE) – Selenium IDE is a record and playback tool. It is distributed as a Firefox Plugin.

 Selenium Remote Control (RC) – Selenium RC is a server that allows user to create test scripts in a desired programming language. It also allows executing test scripts within the large spectrum of browsers.

 Selenium WebDriver – WebDriver is a different tool altogether that has various advantages over Selenium RC. WebDriver directly communicates with the web browser and uses its native compatibility to automate.

 Selenium Grid – Selenium Grid is used to distribute your test execution on multiple platforms and environments concurrently.

7) List out the names of programming languages, browsers and operating systems that are supported by Selenium.

Selenium supports various operating systems, browsers and programming languages. Following is the list:

Programming Languages: C#, Java, Python, PHP, Ruby, Perl, JavaScript.

Operating Systems: Android, iOS, Windows, Linux, Mac, Solaris.

Browsers: Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc

8) What are the significant changes/upgrades in various Selenium versions?

Selenium v1.0:

Version 1.0 was the initial release of Selenium.

It included three tools: Selenium IDE, Selenium RC, and Selenium Grid.

Selenium v2.0:

Selenium WebDriver was introduced replacing Selenium RC in version "2.0".

With the onset of WebDriver, RC got deprecated and moved to the legacy package.

Selenium v3:

The latest release Selenium 3 has new added features and functionalities.

It includes Selenium IDE, Selenium WebDriver, and Selenium Grid.

9) List some of the test types that are supported by Selenium.

Different types of testing's that we can achieve through Selenium are.

Functional Testing: Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user's expectation

Regression Testing: Regression Testing is a type of testing in the software development cycle that runs after every change to ensure that the change introduces no unintended breaks. Regression testing addresses a common issue that developers face — the emergence of old bugs with the introduction of new changes

Sanity Testing: Sanity testing is a subset of regression testing. After receiving the software build, sanity testing is performed to ensure that the code changes introduced are working as expected . This testing is a checkpoint to determine if testing for the build can proceed or not.

Suppose you develop an online food ordering application, and you want to offer a 15% discount on Christmas (25th December) for your members having a premium membership. The sanity testing will verify the following: Premium members get the discount on the 25th.

Smoke Testing: Smoke testing, also called build verification testing or confidence testing, is a software testing method that is used to determine if a new software build is ready for the next testing phase. This testing method determines if the most crucial functions of a program work but does not delve into finer details.

Responsive Testing: Responsive testing is a process that renders web pages on viewports of multiple devices using CSS media queries based on the user device where the website is accessed. In simple terms, responsive testing ensures how responsive web design is optimized well for all types of screen sizes and resolutions.

Cross Browser Testing: Cross browser testing involves comparing and analyzing the behavior of your website in different browser environments. It helps ensure that your website delivers an optimal user experience, independent of the browser used to access it.

UI testing (black box): UI testing or user interface testing is a type of software testing that focuses on checking the appearance, functionality, and usability of different kinds of user interfaces, such as: Graphical user interface (GUI) Command line interface (CLI) Voice user interface (VUI)

Integration Testing: Integration testing involves checking individual components or units of a software project to expose defects and problems to verify that they work together as designed. As a rule, the usual software project consists of numerous software modules, many of them built by different programmers

10. What is Selenium IDE and when should we use it

Selenium IDE is an open source test automation tool that can record and playback your actions on the web. By using it, you can automate tests for web applications. Thanks to its convenient features, manual testers who have to repeat their test scenarios are benefiting from the tool.

Selenium IDE is implemented as Firefox extension which provides record and playback functionality on test scripts. It allows testers to export recorded scripts in many languages like HTML, Java, Ruby, RSpec, Python, C#, JUnit and TestNG.

Selenium IDE has limited scope, and the generated test scripts are not very robust, and portable.

Selenium IDE is the simplest and easiest of all the tools within the Selenium Package. Its record and playback feature makes it exceptionally easy to learn with minimal acquaintances to any programming language. Selenium IDE is an ideal tool for a naïve user

Drawbacks

The Selenium IDE lacks conditional statements, logging and reporting functionality, loops, database testing, and it can not handle exceptions or automatically re-run tests that have failed. It also can't take screenshots. Another downside is that it's Firefox only. If the Selenium IDE is used in the Firefox browser's side-bar, then the quality engineer can't use it to record any actions undertaken by a user in a separate

11) What do you mean by Selenese?

Selenese is the language used to write Selenium Commands. These Selenese commands are then used to test web-applications. Based on the HTML tags of the UI elements, one can check their existence. Commands help Selenium understand what actions or operations to perform.

12) What are the different ways of locating a web element in Selenium?

In Selenium, web elements are identified and located with the help of Locators. Locators specify a target location which uniquely defines the web element in the context of a web application. Thus, to identify web elements accurately and precisely we have different types of locators in Selenium:

In Selenium, locating web elements is a crucial aspect of web automation testing. Web elements such as buttons, text fields, checkboxes, etc., need to be identified and interacted with during test automation. Selenium provides various methods for locating these elements. Here are some common ways to locate web elements:

1. **By ID:**
   - Example in Java:

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

2. **By Name:**
   - Example:

```
WebElement element = driver.findElement(By.name("elementName"));
```

3. **By Class Name:**
   - Example:

```
WebElement element = driver.findElement(By.className("elementClass"));
```

4. **By Tag Name:**
   - Example:

```
WebElement element = driver.findElement(By.tagName("tagName"));
```

5. **By Link Text:**
   - Example:

```
WebElement element = driver.findElement(By.linkText("Link Text"));
```

6. **By Partial Link Text:**

- Example:

```
WebElement element = driver.findElement(By.partialLinkText("Partial Li
nk Text"));
```

7. **By XPath:**

- Example:

```
WebElement element = driver.findElement(By.xpath("//div[@id='exampl
e']"));
```

8. **By CSS Selector:**

- Example:

```
WebElement element = driver.findElement(By.cssSelector("div#exampl
e"));
```

9. **Combining Locators with `and` and `or`:**

- Example:

```
WebElement element = driver.findElement(By.xpath("//input[@id='usernam
e' and @type='text']"));
```

10. **Using `findElements` to locate multiple elements:**

- Example:

```
List<WebElement> elements = driver.findElements(By.className("multiple
Elements"));
```

11. **Dynamic XPath with Contains:**

- Example:

```
WebElement element = driver.findElement(By.xpath("//input[contains(@i
d,'partialId')]"));
```

12. **Dynamic XPath with Starts-with:**

- Example:

```
WebElement element = driver.findElement(By.xpath("//input[starts-with
(@id,'partialId')]"));
```

13) How many types of WebDriver API's are available in Selenium?

The list of WebDriver API's which are used to automate browser include:

AndroidDriver

EventFiringWebDriver

HtmlUnitDriver

iPhoneDriver

iPhoneSimulatorDriver

1. **WebDriver for Chrome (`ChromeDriver`):**

- Used for automation in the Google Chrome browser.

Example in Java:

```
WebDriver driver = new ChromeDriver();
```

2. **WebDriver for Firefox (`FirefoxDriver`):**

- Used for automation in the Mozilla Firefox browser.

Example in Java:

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

3. **WebDriver for Internet Explorer (`InternetExplorerDriver`):**

- Used for automation in the Internet Explorer browser.

Example in Java:

```
WebDriver driver = new InternetExplorerDriver();
```

4. **WebDriver for Edge (`EdgeDriver`):**

- Used for automation in the Microsoft Edge browser.

Example in Java:

```
WebDriver driver = new EdgeDriver();
```

5. **WebDriver for Safari (`SafariDriver`):**

- Used for automation in the Safari browser (primarily on macOS).

Example in Java:

```
WebDriver driver = new SafariDriver();
```

6. **WebDriver for Opera (`OperaDriver`):**

- Used for automation in the Opera browser.

Example in Java:

```
WebDriver driver = new OperaDriver();
```

7. **WebDriver for Remote Execution (`RemoteWebDriver`):**

- Allows you to run your tests on a remote machine (hub) while controlling the browser on a different machine (node).

Example in Java:

```
WebDriver driver = new RemoteWebDriver(new URL("http://hub_address:port/wd/hub"), capabilities);
```

It's important to keep in mind that browser vendors regularly update their browsers, and WebDriver releases may also be updated to match the browser versions. Check the Selenium documentation or the official WebDriver documentation for the latest information on supported browsers and WebDriver versions.

14) List out some of the Automation tools which could be integrated with Selenium to achieve

Selenium can be integrated with automation tools like Maven, Jenkins, &Docker to achieve continuous testing. It can also be integrated with tools such as TestNG, &JUnit for managing test cases and generating reports.

Selenium is a powerful tool for web automation testing, but when it comes to achieving continuous testing as part of a continuous integration/continuous deployment (CI/CD) pipeline, it often needs to be integrated with other tools. Here are some popular automation tools that can be integrated with Selenium for continuous testing:

1. **TestNG:**
    - TestNG is a testing framework for Java inspired by JUnit and NUnit. It supports parallel test execution and provides annotations for test configuration.
2. **JUnit:**
    - JUnit is a widely used testing framework for Java. Selenium can be integrated with JUnit to organize and execute tests.
3. **Maven:**
    - Maven is a build automation and project management tool. It can be used to manage Selenium dependencies, build projects, and execute tests.
4. **Jenkins:**
    - Jenkins is a popular open-source automation server that supports building, deploying, and automating any project, including Selenium tests. Jenkins can be configured to trigger test execution on code changes.
5. **BrowserStack:**
    - BrowserStack is a cloud-based cross-browser testing platform. It enables Selenium tests to be executed on a wide range of browsers and devices.

When setting up continuous testing, the choice of tools depends on the specific requirements of the project, the existing infrastructure, and the team's preferences. The integration of these tools with Selenium can enhance the overall automation testing process and support continuous integration and continuous deployment practices.

15) What do you mean by the assertion in Selenium?

In Selenium, Asserts are validations or checkpoints for an application. Assertions state confidently that application behavior is working as expected. Asserts in Selenium validate the automated test cases that help testers understand if tests have passed or failed. T

Selenium Assertions can be of three types: “assert”, “verify”, and ” waitFor”. When an “assert” fails, the test is aborted. When a “verify” fails, the test will continue execution, logging the failure.

16) Explain the difference between assert and verify commands?

Assert: Assert command checks if the given condition is true or false. If the condition is true, the program control will execute the next phase of testing, and if the condition is false, execution will stop, and nothing will be executed.

import org.testng.Assert; import org.testng.annotations.Test;

public class ExampleTest { @Test public void testExample() { int actual = 5; int expected = 10;

```
        // If this assertion fails, the test will be marked as failed
        Assert.assertEquals(actual, expected, "The values are not equal");
    }

}
```

Verify: Verify command also checks if the given condition is true or false. It doesn't halt program execution, i.e., any failure during verification would not stop the execution, and all the test phases would be executed.

import org.testng.Assert; import org.testng.annotations.Test;

public class ExampleTest { @Test public void testExample() { int actual = 5; int expected = 10;

```
        // This hypothetical "verify" does not stop the test on failure
        if (actual != expected) {
            System.out.println("Verification failed: The values are not equal");
        }

        // Test continues to execute the next steps
        System.out.println("Continuing with the test...");
    }

}
```

17) What do you mean by XPath?

XPath provides a powerful way to locate elements in an HTML document, making it a valuable tool for web automation testing with Selenium.

XPath is commonly used with Selenium WebDriver to locate web elements for interaction in test automation scripts. However, it's important to use XPath judiciously, as overly complex or fragile XPath expressions can make the test scripts less maintainable. When possible, it's a good practice to use other locators like ID, class name, name, etc., before resorting to XPath

XPath is also defined as XML Path. It is a language used to query XML documents. It is an important approach to locate elements in Selenium. XPath consists of a path expression along with some conditions. Here, we can easily write XPath script/query to locate any element in the webpage. It is developed to allow the navigation of XML documents. The key factors that it considered while navigating are selecting individual elements, attributes, or some other part of an XML document for specific processing. It also produces reliable locators. Some other points about XPath are as follows.

XPath is a language used for locating nodes in XML documents.

XPath can be used as a substitute when you don't have a suitable id or name attribute for the element you want to locate.

There are two types of XPath expressions:

XPath Absolute

It starts from the root element and follows the complete path down to the desired element. It begins with a single forward slash / and specifies the complete hierarchy. Example:

/html/body/div[1]/form/input

Relative XPath:

It starts from the current node and can traverse through the document using the available tags and attributes. It does not necessarily begin from the root element.

Example:

//input[@id='username']

18) What is the difference between "/" and "//" in XPath?

In XPath, the symbols "/" and "//" have distinct meanings and are used to navigate through the structure of XML or HTML documents. These symbols play a crucial role in constructing XPath expressions for locating elements. Here's an explanation of the differences between "/" and "//":

1. **Single Slash ("/"):**

   - The single slash ("/") is used to create an absolute XPath expression. It defines a direct path from the root node to the desired node or element. It specifies an exact location in the document hierarchy.

   Example:

   ```
   /html/body/div[1]/p
   ```

   In this example, the XPath expression starts from the root node (`/html`) and navigates down to the first `div` element and then to the first `p` element under that `div` .
2. **Double Slash ("//"):**

   - The double slash ("//") is used to create a relative XPath expression. It selects nodes or elements at any level of the document hierarchy, not just those directly beneath the current node. It allows for a more flexible and general search for elements.

   Example:

   ```
   //div//p
   ```

   In this example, the XPath expression selects all `p` elements that are descendants of any `div` element, regardless of their direct parent-child relationship.

Key Differences:

- **Absolute vs. Relative:**
  - The single slash ("/") represents an absolute path starting from the root, specifying an exact location.
  - The double slash ("//") represents a relative path, allowing for a more flexible and general search for elements at any level.
- **Direct vs. Any Level:**
  - The single slash ("/") selects elements at a specific level in the hierarchy, following a direct parent-child relationship.
  - The double slash ("//") selects elements at any level in the hierarchy, ignoring the direct parent-child relationship.

Choosing Between "/" and "//":

- Use "/" when you want to specify an exact and specific path in the document hierarchy.
- Use "//" when you want to perform a more general search and locate elements at any level in the hierarchy.

In Selenium WebDriver and other web automation scenarios, understanding the use of "/" and "//" in XPath expressions is essential for accurately locating web elements. It's often a matter of choosing the appropriate one based on the structure of the HTML document and the desired level of specificity in the element selection.

19.) What are the different types of annotations

Annotations in Selenium are used to provide additional information about methods, classes, or interfaces during the execution of test scripts. Annotations help in organizing and controlling the flow of test execution. The most commonly used annotations in Selenium are associated with testing frameworks like TestNG and JUnit. Here are some of the key annotations used in Selenium:

1. **@Test:**

   - Indicates that a method is a test method. This annotation is used by testing frameworks to identify and execute test cases.

```
import org.testng.annotations.Test;

public class ExampleTest {
    @Test
    public void testExample() {
        // Test logic goes here
    }
}
```

2. **@BeforeMethod:**

   - Specifies that a method should be executed before each test method. Commonly used for setting up preconditions or initializing resources.

```
import org.testng.annotations.BeforeMethod;

public class ExampleTest {
    @BeforeMethod
    public void setUp() {
        // Code to set up preconditions
    }

    @Test
    public void testExample() {
        // Test logic goes here
    }
}
```

3. **@AfterMethod:**

   - Specifies that a method should be executed after each test method. Commonly used for cleaning up resources or performing post-test actions.

```
import org.testng.annotations.AfterMethod;

public class ExampleTest {
    @Test
    public void testExample() {
        // Test logic goes here
    }

    @AfterMethod
    public void tearDown() {
        // Code to clean up resources
    }
}
```

4. **@BeforeClass:**

- Specifies that a method should be executed before any test methods in a particular class. Used for class-level setup.

```
import org.testng.annotations.BeforeClass;

public class ExampleTest {
    @BeforeClass
    public void classSetup() {
        // Code to be executed before any test method in the class
    }

    @Test
    public void testExample() {
        // Test logic goes here
    }
}
```

5. **@AfterClass:**

- Specifies that a method should be executed after all test methods in a particular class. Used for class-level teardown.

```
import org.testng.annotations.AfterClass;

public class ExampleTest {
    @Test
    public void testExample() {
        // Test logic goes here
    }

    @AfterClass
    public void classTeardown() {
        // Code to be executed after all test methods in the class
    }
}
```

6. **@BeforeSuite:**

- Specifies that a method should be executed before any test in the suite. Used for suite-level setup.

```
import org.testng.annotations.BeforeSuite;

public class ExampleTest {
    @BeforeSuite
    public void suiteSetup() {
        // Code to be executed before any test in the suite
    }

    @Test
    public void testExample() {
        // Test logic goes here
    }
}
```

7. **@AfterSuite:**

- Specifies that a method should be executed after all tests in the suite. Used for suite-level teardown.

```
import org.testng.annotations.AfterSuite;

public class ExampleTest {
    @Test
    public void testExample() {
        // Test logic goes here
    }

    @AfterSuite
    public void suiteTeardown() {
        // Code to be executed after all tests in the suite
    }
}
```

20) What are the WebDriver supported Mobile Testing Drivers?

WebDriver supported "mobile testing drivers" are:

AndroidDriver

IphoneDriver

OperaMobileDriver

Selenium provides WebDriver bindings for mobile testing, allowing automation of mobile web applications on real devices and emulators/simulators. The WebDriver-supported mobile testing drivers include:

1. **AndroidDriver:**

   - AndroidDriver is part of the Selenium WebDriver suite and is used for automating mobile web applications on Android devices. It communicates with the Android device using the Appium framework.

   Example in Java:

```
import io.appium.java_client.android.AndroidDriver;
import java.net.URL;
import org.openqa.selenium.remote.DesiredCapabilities;

public class AndroidExample {
    public static void main(String[] args) throws Exception {
        DesiredCapabilities capabilities = new DesiredCapabilities();
        capabilities.setCapability("deviceName", "Android Emulator");
        capabilities.setCapability("platformName", "Android");
        capabilities.setCapability("browserName", "Chrome");

        AndroidDriver driver = new AndroidDriver(new URL("http://127.0.0.
1:4723/wd/hub"), capabilities);
        // Your mobile web automation logic goes here

        driver.quit();
    }
}
```

2. **IOSDriver:**

   - IOSDriver is used for automating mobile web applications on iOS devices. It communicates with the iOS device using the Appium framework.

Example in Java:

```
import io.appium.java_client.ios.IOSDriver;
import java.net.URL;
import org.openqa.selenium.remote.DesiredCapabilities;

public class IOSExample {
    public static void main(String[] args) throws Exception {
        DesiredCapabilities capabilities = new DesiredCapabilities();
        capabilities.setCapability("deviceName", "iPhone Simulator");
        capabilities.setCapability("platformName", "iOS");
        capabilities.setCapability("browserName", "Safari");

        IOSDriver driver = new IOSDriver(new URL("http://127.0.0.1:4723/w
d/hub"), capabilities);
        // Your mobile web automation logic goes here

        driver.quit();
    }
}
```

These drivers leverage the Appium framework, which extends the WebDriver protocol to support mobile automation for both Android and iOS platforms. Additionally, Appium allows testing of native, hybrid, and mobile web applications. The examples above demonstrate how to use AndroidDriver and IOSDriver for mobile web automation with Selenium and Appium. Keep in mind that the specific capabilities and setup may vary based on the requirements and the environment in which you are conducting mobile testing.

21.What is the difference between type keys and type commands?

TypeKeys() will trigger JavaScript event in most of the cases whereas .type() won't.

22.What is the difference between "type" and "typeAndWait" command?

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

23.What is the difference between findElement() and findElements()?

Selenium WebDriver defines two methods for identifying the elements, they are findElement and findElements . findElement: This command is used to uniquely identify a web element within the web page. findElements: This command is used to uniquely identify the list of web elements within the web page

In Selenium WebDriver, both `findElement()` and `findElements()` are methods used to locate elements on a web page, but they have some key differences:

1. **`findElement()` Method:**
    - `findElement()` is used to locate the first web element within the current context that matches the specified locator strategy. It returns a single WebElement.

- If no matching element is found, it throws a `NoSuchElementException`.
- Commonly used for locating a single unique element on a page.

Example in Java:

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

2. **`findElements()` Method:**

- `findElements()` is used to locate all web elements within the current context that match the specified locator strategy. It returns a list of WebElements.
- If no matching elements are found, it returns an empty list (not null), so there is no exception thrown.
- Commonly used for locating multiple elements based on a common attribute or tag name.

Example in Java:

```
List<WebElement> elements = driver.findElements(By.className("exampleClas
s"));
```

Key Differences:

- **Single Element vs. List of Elements:**
  - `findElement()` returns a single WebElement.
  - `findElements()` returns a List of WebElements.
- **Exception Handling:**
  - `findElement()` throws a `NoSuchElementException` if no matching element is found.
  - `findElements()` does not throw an exception if no matching elements are found; it returns an empty list.
- **Usage Scenario:**
  - Use `findElement()` when you expect only one unique element to match the given criteria.
  - Use `findElements()` when you expect multiple elements to match the given criteria, or when you want to handle scenarios where no elements are found gracefully.

Examples:

Using `findElement()`:

```
// Finding a single element by ID
WebElement singleElement = driver.findElement(By.id("exampleId"));
```

Using `findElements()`:

```
// Finding multiple elements by class name
List<WebElement> multipleElements = driver.findElements(By.className("exampleC
lass"));

// Handling the case where no elements are found
if (multipleElements.isEmpty()) {
    System.out.println("No elements found");
} else {
    // Process the list of elements
    for (WebElement element : multipleElements) {
        // Process each element
    }
}
```

In summary, `findElement()` is used for finding a single element, while `findElements()` is used for finding multiple elements and is more tolerant when no matching elements are found. The appropriate method to use depends on the specific requirements of your test scenario.

24.What is the wait? How many types of waits in selenium?

Selenium provides various wait mechanisms to help wait for an element to appear, disappear, or be clickable. These wait mechanisms can be classified into three types: implicit wait, explicit wait, and fluent wait.

In Selenium, a wait is a mechanism used to pause the execution of a test script for a specified amount of time or until a certain condition is met. Waits are crucial for handling dynamic web pages where elements might load or change state asynchronously. They help ensure that the test script proceeds only when the application is in a stable state, preventing issues related to element visibility, presence, or interaction.

There are two main types of waits in Selenium:

1. **Implicit Wait:**

   - An implicit wait tells the WebDriver to wait for a certain amount of time before throwing a `NoSuchElementException`. It is set once for the entire duration of the WebDriver instance.
   - The implicit wait is applied globally to every findElement() or findElements() method call.

   Example in Java:

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

2. **Explicit Wait:**

   - An explicit wait is more specific and allows the WebDriver to wait for a certain condition to be met before proceeding further in the script. It is applied to a specific WebElement.
   - WebDriverWait is commonly used for explicit waits along with ExpectedConditions, which define the expected conditions to wait for.

   Example in Java:

   ```
   WebElement element = (new WebDriverWait(driver, 10))
       .until(ExpectedConditions.visibilityOfElementLocated(By.id("exampleI
   d")));
   ```

   In this example, the script will wait for a maximum of 10 seconds until the element with ID "exampleId" becomes visible.

Key Differences:

- **Implicit Wait:**
  - Defined once for the entire WebDriver instance.
  - Applied globally to every findElement() or findElements() method call.
  - Waits for a specified maximum amount of time before throwing a `NoSuchElementException`.
- **Explicit Wait:**
  - Defined for a specific WebElement and a specific condition.
  - Applied using WebDriverWait along with ExpectedConditions.
  - Waits for a specified maximum amount of time until the condition is met.

When to Use Each Type:

- Use **Implicit Wait** when you want to set a default waiting time for every element to be found. It's suitable for scenarios where most elements on the page share a similar load time.
- Use **Explicit Wait** when you need more fine-grained control over waiting conditions. It allows you to wait for specific conditions for a particular WebElement, making it more flexible.

25. What is Selenium Grid? and When should I use Selenium Grid?

Selenium Grid: this part of Selenium is used to run tests in different machines simultaneously

Selenium Grid is a smart proxy server that makes it easy to run tests in parallel on multiple machines. This is done by routing commands to remote web browser instances, where one server acts as the hub. This hub routes test commands that are in JSON format to multiple registered Grid nodes.

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, testing under different environments and saving execution time remarkably.

26.How can we launch different browsers in Selenium WebDriver?

We have to create an instance of a driver of that particular browser.

WebDriver driver =newFirefoxDriver();

Here, "WebDriver" is an interface, and we are creating a reference variable "driver" of type WebDriver, instantiated using "FireFoxDriver" class.

Selenium WebDriver allows you to automate web browsers for testing purposes, and it supports various browsers. Here's how you can launch different browsers using Selenium WebDriver in popular programming languages like Java, Python, and C#:

1. Java:

Chrome:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class ChromeExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver.ex
e");
        WebDriver driver = new ChromeDriver();
        // Your automation code here
        driver.quit();
    }
}
```

Firefox:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class FirefoxExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.gecko.driver", "path/to/geckodriver.ex
e");
        WebDriver driver = new FirefoxDriver();
        // Your automation code here
        driver.quit();
    }
}
```

Edge:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.edge.EdgeDriver;

public class EdgeExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.edge.driver", "path/to/msedgedriver.ex
e");
        WebDriver driver = new EdgeDriver();
        // Your automation code here
        driver.quit();
    }
}
```

2. Python:

Chrome:

```
from selenium import webdriver

chrome_path = "path/to/chromedriver.exe"
driver = webdriver.Chrome(executable_path=chrome_path)
# Your automation code here
driver.quit()
```

Firefox:

```
from selenium import webdriver

firefox_path = "path/to/geckodriver.exe"
driver = webdriver.Firefox(executable_path=firefox_path)
# Your automation code here
driver.quit()
```

Edge:

```
from selenium import webdriver

edge_path = "path/to/msedgedriver.exe"
driver = webdriver.Edge(executable_path=edge_path)
# Your automation code here
driver.quit()
```

3. C#:

Chrome:

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

class ChromeExample
{
    static void Main()
    {
        IWebDriver driver = new ChromeDriver("path/to/chromedriver.exe");
        // Your automation code here
        driver.Quit();
    }
}
```

Firefox:

```
using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;

class FirefoxExample
{
    static void Main()
    {
        IWebDriver driver = new FirefoxDriver("path/to/geckodriver.exe");
        // Your automation code here
        driver.Quit();
    }
}
```

Edge:

```
using OpenQA.Selenium;
using OpenQA.Selenium.Edge;

class EdgeExample
{
```

27.Write a code snippet to perform right-click an element in WebDriver.

We will use Action class to generate user event like right-click an element in WebDriver.

Actions action = newActions(driver);

WebElement element = driver.findElement(By.id("elementId"));

action.contextClick(element).perform();

In Selenium WebDriver, you can perform a right-click (context-click) on an element using the `Actions` class. Here's a code snippet in Java as an example:

```
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.interactions.Actions;

public class RightClickExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver.ex
e");
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com");

        // Locate the element on which you want to perform right-click
        WebElement elementToRightClick = driver.findElement(By.id("exampleElem
entId"));

        // Create an Actions object
        Actions actions = new Actions(driver);

        // Perform right-click on the element
        actions.contextClick(elementToRightClick).perform();

        // Your automation code here (perform any actions you want in the cont
ext menu)

        // Close the browser
        driver.quit();
    }
}
```

This example assumes you have a specific element on a webpage with the id "exampleElementId" that you want to right-click. Replace it with the appropriate locator strategy for your case.

Remember to replace `"path/to/chromedriver.exe"` with the actual path to your ChromeDriver

28.Write a code snippet to perform mouse hover in WebDriver.

In Selenium WebDriver, you can perform a mouse hover action using the `Actions` class. Here's a code snippet in Java as an example:

```
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.interactions.Actions;

public class MouseHoverExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver.ex
e");
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com");

        // Locate the element on which you want to perform mouse hover
        WebElement elementToHover = driver.findElement(By.id("exampleElementI
d"));

        // Create an Actions object
        Actions actions = new Actions(driver);

        // Perform mouse hover on the element
        actions.moveToElement(elementToHover).perform();
```

29.How do you perform drag and drop operation in WebDriver?

Performing a drag and drop operation in Selenium WebDriver can be accomplished using the `Actions` class. Here's a code snippet in Java as an example:

```
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.interactions.Actions;

public class DragAndDropExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver.ex
e");
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com");

        // Locate the source element (element to be dragged)
        WebElement sourceElement = driver.findElement(By.id("sourceElementI
d"));

        // Locate the target element (element where the source element will be
dropped)
        WebElement targetElement = driver.findElement(By.id("targetElementI
d"));

        // Create an Actions object
        Actions actions = new Actions(driver);
```

30.What are the different methods to refresh a web page in WebDriver?

There are multiple ways of refreshing a page in Webdriver.

1. Using driver.navigate command -

driver.navigate().refresh();

2. Using driver.getCurrentUrl() with driver.get() command -

driver.get(driver.getCurrentUrl());

3. Using driver.getCurrentUrl() with driver.navigate() command -

driver.navigate().to(driver.getCurrentUrl());

4. Pressing an F5 key on any textbox using the sendKeys command -

driver.findElement(By textboxLocator).sendKeys(Keys.F5);

5. Passing ascii value of the F5 key, i.e., "\uE035" using the sendKeys command -

driver.findElement(By textboxLocator).sendKeys("\uE035");

31.Write a code snippet to navigate back and forward in browser history?

Navigate back in browser history:

driver.navigate().back();

Navigate forward in browser history:

driver.navigate().forward();

In Selenium WebDriver, you can navigate back and forward in the browser history using the `navigate()` method provided by the `WebDriver` interface. Here's a code snippet in Java as an example:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class NavigationExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver.ex
e");
        WebDriver driver = new ChromeDriver();

        // Navigate to a website
        driver.get("https://example.com");

        // Perform some actions on the first page
        // ...

        // Navigate to a different page
        driver.get("https://example.com/second-page");

        // Perform some actions on the second page
        // ...

        // Navigate back to the first page
        driver.navigate().back();

        // Perform some actions again on the first page
        // ...

        // Navigate forward to the second page
        driver.navigate().forward();

        // Perform some actions again on the second page
        // ...

        // Close the browser
        driver.quit();
    }
}
```

This example uses the `navigate().back()` method to go back in the browser history and `navigate().forward()` to go forward. Note that these methods simulate the user's navigation through the browser history, so you can use them to revisit previously visited pages.

Remember to replace `"path/to/chromedriver.exe"` with the actual path to your ChromeDriver executable. Also, make sure your WebDriver version matches the browser version installed on your machine.

32.How to invoke an application in WebDriver?

driver.get("url"); or

driver.navigate().to("url");

33.What are the benefits of Automation Testing?

Benefits of Automation testing are as follows.

It allows execution of repeated test cases

It enables parallel execution

Automation Testing encourages unattended execution

It improves accuracy. Thus, it reduces human-generated errors

It saves time and money.

Automation testing offers numerous benefits that contribute to the efficiency, effectiveness, and overall success of the software development and testing process. Some key advantages of automation testing include:

1. **Reusability:** Automated test scripts can be reused across different test cases, allowing for efficient test coverage and reducing redundancy in testing efforts.
2. **Efficiency:** Automation allows for faster execution of test cases compared to manual testing. This leads to quicker feedback on the quality of the software, enabling faster releases.
3. **Consistency:** Automated tests perform the same steps and checks consistently, eliminating the potential for human errors that may occur during repetitive manual testing.
4. **Parallel Execution:** Automation tools can execute tests concurrently on multiple environments and devices, enabling parallel testing. This accelerates the testing process, especially for applications that need to support various platforms.
5. **Coverage:** Automation facilitates the execution of a large number of complex test cases during each test cycle. This helps achieve higher test coverage, ensuring that a wide range of scenarios are tested thoroughly.
6. **Regression Testing:** Automated tests are ideal for regression testing, as they quickly and accurately verify that new code changes have not adversely affected existing functionalities. This is especially valuable in agile development environments with frequent releases.
7. **Continuous Integration/Continuous Deployment (CI/CD):** Automation integrates seamlessly with CI/CD pipelines, allowing for the automatic execution of tests after each code commit. This ensures that the codebase remains stable throughout the development process.
8. **Cost Savings:** While there is an initial investment in setting up automation, the long-term benefits include reduced testing time, increased test coverage, and decreased manual testing efforts, leading to overall cost savings.
9. **Data-Driven Testing:** Automation tools support data-driven testing, enabling the testing of multiple sets of input data to validate the behavior of an application under various conditions.
10. **Improved Accuracy:** Automated tests precisely follow predefined steps and compare results against expected outcomes, reducing the likelihood of human errors and ensuring accurate test results.
11. **Early Detection of Defects:** Automated tests can be executed early in the development cycle, allowing for the early detection and resolution of defects. This contributes to improved software quality.
12. **Scalability:** Automation is easily scalable to handle a large number of test cases, making it suitable for projects of varying sizes and complexities.
13. **Enhanced Test Reporting:** Automation tools often provide detailed test reports and logs, making it easier to identify issues and troubleshoot failures. This aids in effective communication within the development and testing teams.

34.How can we get a text of a web element?

getText() Method in Selenium This method helps retrieve the text, which is basically the innertext of a WebElement. getText() method returns string as a result

Get command is used to get the inner text of the specified web element. The get command doesn't require any parameter, but it returns a string type value. It is also one of the widely used commands for verification of messages, labels, and errors,etc.,from web pages.

Syntax

String Text = driver.findElement(By.id("Text")).getText();

To retrieve the text content of a web element using Selenium WebDriver, you can use the `getText()` method. This method is available for various types of web elements, such as `WebElement`, `WebElementList`, etc. Here's an example in Java:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class GetTextExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver.ex
e");
        WebDriver driver = new ChromeDriver();

        // Navigate to a webpage
        driver.get("https://example.com");

        // Locate the web element by its identifier (e.g., ID, XPath, CSS sele
ctor)
        WebElement element = driver.findElement(By.id("exampleElementId"));

        // Get the text of the web element
        String elementText = element.getText();

        // Print or use the retrieved text
        System.out.println("Text of the element: " + elementText);

        // Close the browser
        driver.quit();
    }
}
```

The `getText()` method retrieves the visible text of the web element, excluding any hidden or non-visible text. It's commonly used to verify the text content of elements, extract data, or perform assertions in test automation scripts.

35.How to select value in a dropdown?

We use the WebDriver's Select class to select the value in the dropdown.

Syntax:

selectByValue:

Select selectByValue = new Select(driver.findElement(By.id("SelectID_One")));
selectByValue.selectByValue("greenvalue");

selectByVisibleText:

Select selectByVisibleText = new Select (driver.findElement(By.id("SelectID_Two")));
selectByVisibleText.selectByVisibleText("Lime");

Select selectByIndex = new Select(driver.findElement(By.id("SelectID_Three")));
selectByIndex.selectByIndex(2);

36.What are the different types of navigation commands?

In Selenium WebDriver, navigation commands are used to move between different pages or interact with the browser's history. The primary navigation commands are part of the `WebDriver.Navigation` interface. Here are the main navigation commands in Selenium WebDriver:

1. **`to(String url) / get(String url) :`**
   - `driver.navigate().to("url")` or `driver.get("url")` is used to navigate to the specified URL.
   - These commands are functionally equivalent and are used to open a new URL in the browser.

   ```
   driver.navigate().to("https://example.com");
   ```

2. **`back() :`**
   - `driver.navigate().back()` is used to navigate back to the previous page in the browser history.

   ```
   driver.navigate().back();
   ```

3. **`forward() :`**
   - `driver.navigate().forward()` is used to navigate forward to the next page in the browser history after navigating back.

   ```
   driver.navigate().forward();
   ```

4. **`refresh() :`**
   - `driver.navigate().refresh()` is used to refresh the current page.

   ```
   driver.navigate().refresh();
   ```

5. **`toAlert() :`**
   - `driver.switchTo().alert()` is used to switch to the currently active alert dialog for interacting with it.

   ```
   Alert alert = driver.switchTo().alert();
   ```

6. **`toFrame() :`**
   - `driver.switchTo().frame(int index)` or `driver.switchTo().frame(String nameOrId)` or `driver.switchTo().frame(WebElement frameElement)` is used to switch to a frame or iframe on the page.

```
// By index
driver.switchTo().frame(0);

// By name or ID
driver.switchTo().frame("frameNameOrId");

// By WebElement
WebElement frameElement = driver.findElement(By.id("frameId"));
driver.switchTo().frame(frameElement);
```

37.How to deal with frame in WebDriver?

Handling frames (iframes) in Selenium WebDriver involves switching the context to the frame before performing any actions on elements inside it. Here are the steps to deal with frames in WebDriver:

1. **Switch to a Frame by Index:**

```
driver.switchTo().frame(0); // Switch to the first frame (index starts fro
m 0)
```

2. **Switch to a Frame by Name or ID:**

```
driver.switchTo().frame("frameNameOrId");
```

3. **Switch to a Frame by WebElement:**

```
WebElement frameElement = driver.findElement(By.id("frameId"));
driver.switchTo().frame(frameElement);
```

4. **Switch Back to the Default Content:**

```
driver.switchTo().defaultContent();
```

After switching to a frame, you can perform actions on elements within that frame. Once you're done working inside the frame, it's important to switch back to the default content before interacting with elements outside of any frames.

Here's an example that demonstrates switching to a frame, performing actions, and then switching back to the default content:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class FrameHandlingExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver.ex
e");
        WebDriver driver = new ChromeDriver();

        // Navigate to a webpage with a frame
        driver.get("https://example.com");

        // Switch to the frame by index, name, or ID
        driver.switchTo().frame(0); // or driver.switchTo().frame("frameNameOr
Id");
```

38.How can you redirect browsing from a browser through some proxy?

To redirect browsing through a proxy in Selenium WebDriver, you can use the `Proxy` class along with the `DesiredCapabilities` class. The `Proxy` class allows you to configure proxy settings, and the `DesiredCapabilities` class helps in setting up the desired capabilities for the browser.

Here's an example in Java using ChromeDriver:

```
import org.openqa.selenium.Proxy;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.CapabilityType;

public class ProxyExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver.ex
e");

        // Set proxy settings
        String proxyAddress = "your_proxy_address";
        int proxyPort = 8080;

        Proxy proxy = new Proxy();
        proxy.setHttpProxy(proxyAddress + ":" + proxyPort);
        proxy.setSslProxy(proxyAddress + ":" + proxyPort);
```

39.What is POM (Page Object Model)? What are its advantages??

Page Object Model, also known as POM, is a design pattern in Selenium that creates an object repository for storing all web elements. It helps reduce code duplication and improves test case maintenance.

Page Object Model (POM) is a design pattern used in test automation to create an object-oriented structure for the representation of a web application's pages. In POM, each page of the application is represented as a separate class, and the interaction with the elements on the page is encapsulated within that class. This pattern promotes code reusability, maintainability, and readability in automated test scripts.

Key components of the Page Object Model include:

1. **Page Classes:** Each page of the web application is represented as a separate class. The class contains web elements and methods to interact with those elements.
2. **Test Classes:** Test scripts interact with the web pages by creating instances of the corresponding page classes. This separation between test logic and page interactions enhances code organization and maintainability.
3. **Page Factory:** Page Factory is a concept often used with POM. It is an initialization approach that helps in initializing the web elements defined in a page class.

Advantages of Page Object Model (POM):

1. **Code Reusability:** Page classes encapsulate the behavior and elements of a page. Test scripts can reuse these classes, reducing duplication of code.
2. **Maintainability:** Changes to the application's UI or functionality can be localized to the corresponding page class. If there are updates or modifications, you only need to update the affected page class, making maintenance more straightforward.
3. **Readability:** POM enhances the readability of test scripts by providing a clear and intuitive structure. Test scripts focus on high-level actions, while page classes handle the low-level details of element interaction.
4. **Scalability:** As the application grows, POM allows for easy scaling by adding new page classes. Each new feature or page can be represented by a separate class, making the automation framework scalable and adaptable.
5. **Reduced Code Redundancy:** With the separation of concerns, the code redundancy is minimized. Shared elements and methods are centralized in page classes, reducing the likelihood of duplicated code across test scripts.

6. **Improved Collaboration:** POM encourages a modular structure, making it easier for multiple team members to collaborate on test automation. Testers can work on test scripts, while developers focus on maintaining and updating the page classes.
7. **Enhanced Debugging:** In case of test failures, POM makes it easier to identify the source of the issue. Debugging is more efficient because the code related to a specific page is contained within its dedicated class.
8. **Parallel Execution:** POM facilitates parallel execution of tests since each test script is independent of others and interacts with pages through well-defined interfaces.

By adopting the Page Object Model, teams can build a robust and maintainable test automation

40.How to capture screenshot in WebDriver?

In Selenium WebDriver, you can capture screenshots using the `TakesScreenshot` interface, which is implemented by most WebDriver classes. Below is a basic example using Java and ChromeDriver:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.TakesScreenshot;
import java.io.File;
import org.apache.commons.io.FileUtils;

public class ScreenshotExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver.ex
e");
        WebDriver driver = new ChromeDriver();

        // Navigate to a webpage
        driver.get("https://example.com");

        // Capture screenshot as File
        File screenshotFile = ((TakesScreenshot) driver).getScreenshotAs(org.o
penqa.selenium.OutputType.FILE);

        try {
            // Save the screenshot to a specified location
            FileUtils.copyFile(screenshotFile, new File("path/to/screenshot.pn
g"));
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Close the browser
        driver.quit();
    }
}
```

Here's a breakdown of the code:

- `TakesScreenshot` is an interface provided by Selenium WebDriver for capturing screenshots.
- `((TakesScreenshot) driver).getScreenshotAs(org.openqa.selenium.OutputType.FILE)` captures the screenshot as a `File` object.
- The Apache Commons IO library (`FileUtils.copyFile()`) is used to copy the screenshot file to a specified location.

41.How to type text in a textbox using Selenium?

The sendKeys("String to be entered") is used to enter the string in a textbox.

Syntax

```
WebElement username = drv.findElement(By.id("Email"));
// entering username
username.sendKeys("sth");
```

42.How can you find if an element is displayed on the screen?

WebDriver allows user to check the visibility of the web elements. These web elements can be buttons, radio buttons, drop, checkboxes, boxes, labels etc. which are used with the following methods.

isDisplayed()

isSelected()

isEnabled()

Syntax:

isDisplayed():

```
boolean buttonPresence = driver.findElement(By.id("gbqfba")).isDisplayed();
```

isSelected():

```
boolean buttonSelected = driver.findElement(By.id("gbqfba")).isSelected();
```

isEnabled():

```
boolean searchIconEnabled = driver.findElement(By.id("gbqfb")).isEnabled();
```

43. How to click on a hyper link using linkText?

```
driver.findElement(By.linkText("Google")).click();
```

The above command search the element using a link text, then click on that element and thus the user will be re-directed to the corresponding page.

The following command can access the link mentioned earlier.

```
driver.findElement(By.partialLinkText("Goo")).click();
```

44.Why should Selenium be selected as a test tool?

Selenium

1. is free and open source
2. have a large user base and helping communities
3. have cross Browser compatibility (Firefox, chrome, Internet Explorer, Safari etc.)

4. have great platform compatibility (Windows, Mac OS, Linux etc.)
5. supports multiple programming languages (Java, C#, Ruby, Python, Pearl etc.)
6. has fresh and regular repository developments
7. supports distributed testing

45.What are the limitations of Selenium?

Following are the limitations of Selenium:

 Selenium supports testing of only web based applications

 Mobile applications cannot be tested using Selenium

 Captcha and Bar code readers cannot be tested using Selenium

 Reports can only be generated using third party tools like TestNG or Junit.

 As Selenium is a free tool, thus there is no ready vendor support though the

user can find numerous helping communities.

 User is expected to possess prior programming language knowledge.

46.What is Same origin policy and how it can be handled?

The problem of same origin policy disallows to access the DOM of a document from an origin that is different from the origin we are trying to access the document.

Origin is a sequential combination of scheme, host and port of the URL. For example, for a URL http:// http://www.softwaretestinghelp.com/resources/ (http://www.softwaretestinghelp.com/resources/), the origin is a combination of http, softwaretestinghelp.com, 80 correspondingly.

Thus the Selenium Core (JavaScript Program) cannot access the elements from an origin that is different from where it was launched. For Example, if I have launched the JavaScript Program from "http://www.softwaretestinghelp.com" (http://www.softwaretestinghelp.com%E2%80%9D), then I would be able to access the pages within the same domain such as "http://www.softwaretestinghelp.com/resources" (http://www.softwaretestinghelp.com/resources%E2%80%9D) or "http://www.softwaretestinghelp.com/istqb-free-updates/" (http://www.softwaretestinghelp.com/istqb-free-updates/%E2%80%9D). The other domains like google.com, seleniumhq.org would no more be accessible.

So, In order to handle same origin policy, Selenium Remote Control was introduced.

47.What do we mean by Selenium 1 and Selenium 2?

Selenium RC and WebDriver, in a combination are popularly known as Selenium 2. Selenium RC alone is also referred as Selenium 1.

48.How to find more than one web element in the list?

At times, we may come across elements of same type like multiple hyperlinks, images etc arranged in an ordered or unordered list. Thus, it makes absolute sense to deal with such elements by a single piece of code and this can be done using WebElement List.

// Storing the list List elementList = driver.findElements(By.xpath("//div[@id='example']//ul//li"));

from selenium import webdriver

Assuming you have a WebDriver instance, e.g., ChromeDriver

driver = webdriver.Chrome()

Open a website

driver.get("https://example.com" (https://example.com"))

Find multiple elements based on a specific criteria

elements = driver.find_elements_by_css_selector("your-css-selector")

Iterate through the list of elements and perform actions

for element in elements: # Do something with each element, for example, print its text print(element.text)

Close the browser window

driver.quit()

49.What is the difference between driver.close() and driver.quit command?

close(): WebDriver's close() method closes the web browser window that the user is currently working on or we can also say the window that is being currently accessed by the WebDriver. The command neither requires any parameter nor does is return any value.

quit(): Unlike close() method, quit() method closes down all the windows that the program has opened. Same as close() method, the command neither requires any parameter nor does is return any value.

50. Can Selenium handle windows based pop up?

Selenium is an automation testing tool which supports only web application testing. Therefore, windows pop up cannot be handled using Selenium.

However Selenium alone can't help the situation but along with some third party intervention, this problem can be overcome. There are several third party tools available for handling window based pop ups along with the selenium like AutoIT, Robot class etc

51.How can we handle web based pop up?

WebDriver offers the users with a very efficient way to handle these pop ups using Alert interface. There are the four methods that we would be using along with the Alert interface.

 void dismiss() – The accept() method clicks on the “Cancel” button as soon as the pop up window appears.

 void accept() – The accept() method clicks on the “Ok” button as soon as the pop up window appears.

 String getText() – The getText() method returns the text displayed on the alert box.

 voidsendKeys(String stringToSend) – The sendKeys() method enters the specified string pattern into the alert box.

Syntax: // accepting javascript alert Alertalert = driver.switchTo().alert(); alert.accept();

from selenium import webdriver from selenium.webdriver.common.alert import Alert

driver = webdriver.Chrome() driver.get("https://www.example.com" (https://www.example.com"))

Trigger a prompt pop-up driver.execute_script("prompt('Enter your name:');")

Switch to the alert, provide input, and accept it alert = Alert(driver) alert.send_keys("John Doe") alert.accept()

Close the browser window driver.quit()

52. How to assert title of the web page

//verify the title of the web page assertTrue(“The title of the window is incorrect.”,driver.getTitle().equals(“Title of the page”));

from selenium import webdriver

Create a WebDriver instance (e.g., ChromeDriver) driver = webdriver.Chrome()

Open a web page driver.get("https://www.example.com" (https://www.example.com"))

#Get the title of the web page actual_title = driver.title

#Assert that the actual title matches the expected title expected_title = "Example Domain" assert actual_title == expected_title, f"Title mismatch. Expected: {expected_title}, Actual: {actual_title}"

#If the assertion passes, the script continues; otherwise, it raises an AssertionError

#Close the browser window driver.quit()

53. How to capture screenshot in WebDriver?

importorg.junit.After;

importorg.junit.Before;

importorg.junit.Test;

importjava.io.File;

importjava.io.IOException;

```
importorg.apache.commons.io.FileUtils;

importorg.openqa.selenium.OutputType;

importorg.openqa.selenium.TakesScreenshot;

importorg.openqa.selenium.WebDriver;

importorg.openqa.selenium.firefox.FirefoxDriver;

publicclassCaptureScreenshot

WebDriver driver;

@Before

publicvoidsetUp() throwsException {

driver = newFirefoxDriver();

driver.get("https://google.com" (https://google.com"));

}

@After

publicvoidtearDown() throwsException {

driver.quit();

}

@Test

publicvoidtest() throwsIOException {

// Code to capture the screenshot

File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

// Code to copy the screenshot in the desired location

FileUtils.copyFile(scrFile, newFile("C:\CaptureScreenshot\google.jpg"));

}

}
```

54.What is Junit?

Junit is a unit testing framework introduced by Apache. Junit is based on Java.

JUnit has been influential in the development of testing frameworks for various programming languages and has become a standard for unit testing in Java projects

55.What are Junit annotations?

Following are the Junit Annotations:

 @Test: Annotation lets the system know that the method annotated as @Test is a test method. There can be multiple test methods in a single test script.

 @Before: Method annotated as @Before lets the system know that this method shall be executed every time before each of the test method.

 @After: Method annotated as @After lets the system know that this method shall be executed every time after each of the test method.

 @BeforeClass: Method annotated as @BeforeClass lets the system know that this method shall be executed once before any of the test method.

 @AfterClass: Method annotated as @AfterClass lets the system know that this method shall be executed once after any of the test method.

56.What is TestNG and how is it better than Junit?

TestNG is an advance framework designed in a way to leverage the benefits by both the developers and testers. With the commencement of the frameworks, JUnit gained an enormous popularity across the Java applications, Java developers and Java testers with remarkably increasing the code quality. Despite being easy to use and straightforward, JUnit has its own limitations which give rise to the need of bringing TestNG into the picture. TestNG is an open source framework which is distributed under the Apache software License and is readily available for download.

TestNG with WebDriver provides an efficient and effective test result format that can in turn be shared with the stake holders to have a glimpse on the product's/application's health thereby eliminating the drawback of WebDriver's incapability to generate test reports. TestNG has an inbuilt exception handling mechanism which lets the program to run without terminating unexpectedly.

There are various advantages that make TestNG superior to JUnit. Some of them are:

 Added advance and easy annotations

 Execution patterns can set

 Concurrent execution of test scripts

 Test case dependencies can be set

57.How to set test case priority in TestNG

Setting Priority in TestNG Code Snippet packageTestNG;

importorg.testng.annotations.*;

publicclassSettingPriority {

@Test(priority=0)

publicvoidmethod1() {

}

@Test(priority=1)

publicvoidmethod2() {

}

@Test(priority=2

) publicvoidmethod3() {

```
}

}
```

Test Execution Sequence:

1. Method1
2. Method2
3. Method3

58.What is a framework?

Framework is a constructive blend of various guidelines, coding standards, concepts, processes, practices, project hierarchies, modularity, reporting mechanism, test data injections etc. to pillar automation testing.

59.What are the different types of frameworks?

Below are the different types of frameworks:

1. Module Based Testing Framework: The framework divides the entire “Application Under Test” into number of logical and isolated modules. For each module, we create a separate and independent test script. Thus, when these test scripts taken together builds a larger test script representing more than one module.
2. Library Architecture Testing Framework: The basic fundamental behind the framework is to determine the common steps and group them into functions under a library and call those functions in the test scripts whenever required.
3. Data Driven Testing Framework: Data Driven Testing Framework helps the user segregate the test script logic and the test data from each other. It lets the user store the test data into an external database. The data is conventionally stored in “Key-Value” pairs. Thus, the key can be used to access and populate the data within the test scripts.
4. Keyword Driven Testing Framework: The Keyword driven testing framework is an extension to Data driven Testing Framework in a sense that it not only segregates the test data from the scripts, it also keeps the certain set of code belonging to the test script into an external data file.
5. Hybrid Testing Framework: Hybrid Testing Framework is a combination of more than one above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.
6. Behavior Driven Development Framework: Behavior Driven Development framework allows automation of functional validations in easily readable and understandable format to Business Analysts, Developers, Testers, etc.

60.How can I read test data from excels?

Test data can efficiently be read from excel using JXL or POI API.

61. What is the difference between POI and jxl jar?

JXL jar POI jar

1 JXL supports “.xls” format i.e. binary based format. JXL doesn’t support Excel 2007 and “.xlsx” format i.e. XML based format

POI jar supports all of these formats

2 JXL API was last updated in the year 2009

POI is regularly updated and released

3

The JXL documentation is not as comprehensive as that of POI

POI has a well prepared and highly comprehensive documentation

4 JXL API doesn't support rich text formatting

POI API supports rich text formatting

5 JXL API is faster than POI API

POI API is slower than JXL API

62.What is the difference between Selenium and QTP?

Feature	Selenium	Quick Test Professional (QTP)
Browser Compatibility	Selenium supports almost all the popular browsers like Firefox, Chrome, Safari, Internet Explorer, Opera etc	QTP supports Internet Explorer, Firefox and Chrome. QTP only supports Windows Operating System
Distribution	Selenium is distributed as an open source tool and is freely available	QTP is distributed as a licensed tool and is commercialized
Application under Test	Selenium supports testing of only web based applications	QTP supports testing of both the web based application and windows based application
Object Repository	Object Repository needs to be created as a separate entity	QTP automatically creates and maintains Object Repository
Language Support	Selenium supports multiple programming languages like Java, C#, Ruby, Python, Perl etc	QTP supports only VB Script
Vendor Support	As Selenium is a free tool, user would not get the vendor's support in troubleshooting issues	Users can easily get the vendor's support in case of any issue

63. Can WebDriver test Mobile applications?

WebDriver cannot test Mobile applications. WebDriver is a web based testing tool, therefore applications on the mobile browsers can be tested.

64. Can captcha be automated?

No, captcha and bar code reader cannot be automated.

65. What is Object Repository? How can we create Object Repository in Selenium?

Object Repository is a term used to refer to the collection of web elements belonging to Application Under Test (AUT) along with their locator values. Thus, whenever the element is required within the script, the locator value can be populated from the Object Repository. Object Repository is used to store locators in a centralized location instead of hard coding them within the scripts.

In Selenium, objects can be stored in an excel sheet which can be populated inside the script whenever required.

That’s all for now. Hope in this article you will find answers to most frequently asked Selenium and WebDriver Interview questions. The answers provided here are also helpful for understanding the Selenium basics and advanced WebDriver topics.

66.Explain some disadvantages to manual software testing.

Manual software testing takes huge amounts of time and resources, both human and machine. It’s a potentially exhausting process that can end up costing more time and money for the company than if the process was simply automated, owing to employee fatigue and its consequences: inaccuracy, missed issues, lack of clarity.

67.Does automation testing have any disadvantages?

Designing the tools and tests to run software through takes a lot of manual, human effort, though there are frameworks and tests ready made for engineers to use. Even with automated testing, human error is still a factor – tools can be buggy, inefficient, costly, and sometimes even technologically limited in what kinds of tests they can run on their own.

68.What are the differences between open source tools, vendor tools, and in□house tools?

Open source tools are free to use frameworks and applications. Engineers build the tool, and have the source code available for free on the internet for other engineers to use.

Vendor tools are developed by companies that come with licenses to use, and often cost money. Because they are developed by an outside source, technical support is often available for use. Example vendor tools include WinRunner, SilkTest, Rational Robot, QA Director, QTP, LR, QC, RFT, and RPT.

An in-house tool is a tool that a company builds for their own use, rather than purchasing vendor tools or using open source tools.

69.How do you choose which automation tool is best for your specific scenario?

In order to choose the proper automation testing tool, you must consider:

□ the scope of the project

□ the limitation of the tool

□ how much the tool costs

□ the tool’s usability and convenience

 the testing environment

 compatibility

70.How can we get the font size, font colour, font text used for the particular text on the web page use in the selenium?

By using getCCSValue(“font-size”);

It’s like that driver.findElement(By.id()).getCCSValue(“font-size”);

It’s like that driver.findElement(By.id()).getCCSValue(“font-colour”);

It’s like that driver.findElement(By.id()).getCCSValue(“font-type”);

It’s like that driver.findElement(By.id()).getCCSValue(“background-colour”);

71.What is the difference between driver.get and driver.navigate().to(“URL”)?

Both does the same thing but navigate also have some another methods like back(), refresh(), forward(),

In Selenium WebDriver, both `driver.get("URL")` and `driver.navigate().to("URL")` are used to navigate to a specified URL, but there are some subtle differences:

1. **`driver.get("URL")` :**
   - This method is part of the `WebDriver` interface.
   - It is a convenient method that is a shorthand for `driver.navigate().to("URL")` .
   - It is a synchronous method, meaning it will wait for the page to load completely before moving on to the next line of code.
   - It's often used for straightforward navigation to a URL.

   Example:

   ```
   WebDriver driver = new ChromeDriver();
   driver.get("https://www.example.com");
   ```

2. **`driver.navigate().to("URL")` :**
   - This method is part of the `Navigation` interface in Selenium.
   - It provides additional navigation options beyond just opening a URL, such as navigating forward, backward, refreshing, etc.
   - It is asynchronous, meaning it doesn't wait for the page to load completely before moving on to the next line of code.
   - It's useful when you want to navigate to a URL but don't need to wait for the page to fully load before performing the next action.

   Example:

   ```
   WebDriver driver = new ChromeDriver();
   driver.navigate().to("https://www.example.com");
   ```

72. How to clear cache using selemium?

There is multiple ways to delete the cookies

1: driver.manage().deleteAllCookies();

2: If above command does not work then use this DesiredCapabilities capabilities = DesiredCapabilities.internetExplorer();
capabilities.setCapability(InternetExplorerDriver.IE_ENSURE_CLEAN_SESSION, true);

73.How to install add on the Firefox?

Using the firefox profile we can add the extension

packagecom.helloselenium.selenium.test;

importjava.io.File;

importorg.openqa.selenium.WebDriver;

importorg.openqa.selenium.firefox.FirefoxDriver;

importorg.openqa.selenium.firefox.FirefoxProfile;

importorg.openqa.selenium.remote.DesiredCapabilities;

publicclassRunFirefoxWithAddons{

publicstaticvoidmain(String[] args) {

WebDriver driver = null;

FirefoxProfile firefoxProfile = newFirefoxProfile();

File addonpath = newFile("path of addon/extension (.xpi file)");

firefoxProfile.addExtension(addonpath);

DesiredCapabilities capabilities = DesiredCapabilities.firefox();

capabilities.setCapability(FirefoxDriver.PROFILE, profile);

driver = newFirefoxDriver(capabilities);

driver.get("http://www.helloselenium.com" (http://www.helloselenium.com"));

driver.quit();

}

}

74.HOW TO RUN SELENIUM WEBDRIVER SCRIPT IN FIREFOX BROWSER USING DESIREDCAPABILITIES?

First get the desiredCapabilities and then set

packagecom.helloselenium.selenium.test;

```
importorg.openqa.selenium.WebDriver;

importorg.openqa.selenium.firefox.FirefoxDriver;

importorg.openqa.selenium.remote.DesiredCapabilities;

publicclassOpenHelloSeleniumBlogInFirefox{

publicstaticvoidmain(String[] args) {

    ebDriver driver = null;

DesiredCapabilities capabilities = DesiredCapabilities.firefox();

capabilities.setCapability(capability arg1, capability arg2);

driver = newFirefoxDriver(capabilities);

driver.get("http://www.helloselenium.com" (http://www.helloselenium.com"));

driver.quit();

}

}
```

75.How to read data from the .properties files?

```
package framework.vtiger.UtilMethods;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.Properties;

public class ConfigFile {

public static void main(String[] args) throws IOException {

fn_ConfigFileRead("ExcelSheet\config.properties");

}

public static void fn_ConfigFileRead(String propertyFile) throws IOException{

Properties propObj = new Properties();

FileInputStream fisObj = new FileInputStream(propertyFile);

propObj.load(fisObj);

String uname = propObj.getProperty("username");

System.out.println(uname);

String uPassword = propObj.getProperty("password");

System.out.println(uPassword);

}

}
```

76. How to Kill all the browser at the same time?

We are using the .bat file in this we used “taskkill” command which work on the cmd

taskkill /F /IM IEDriverServer.exe /T

taskkill /F /IM chromedriver.exe /T

taskkill /F /IM iexplore.exe /T

taskkill /F /IM firefox.exe /T

exit

77.What is the difference betweet explicitly wait and implicitly wait?

There are primarily two types of waits available in Selenium WebDriver.

– Implicit Wait

– Explicit Wait

Implicit Wait

Implicit waits are used to provide the latency within each and every test step of the test script. Thus, the program control would wait for the specified time before moving the execution control to the next step. Thus the implicit waits are applied to all the test step of the testscript. In other words we can say that the system would wait for the specified period of time in order to load the desired object into the DOM.

//Create WebDriver instance variable

WebDriver driver;

//Launch browser

driver=new FirefoxDriver();

//Apply implicit wait

driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

Explicit Wait

Explicit waits are smarter waits than implicit waits. Explicit waits can be applied at certain instances instead of applying on every web element within the testscript. Suppose if we are creating a login script for Gmail. We know that we are most likely to wait for a few seconds to let the home page load successfully. Thus, in such cases explicit waits can be used. Another important benefit that we get from explicit wait is that it waits maximum for the specified period of time or a condition to be met. Thus, if our condition (Let’s say an element should be visible before we click on it) ismet before the specified time has elapsed, then the system control would move forward rather than waiting for the complete time to elapse in order to save our execution time.

Code Sample

public void explicitWait(WebDriver driver)

{

WebDriverWait wait = new WebDriverWait(driver, 20);

wait.until(ExpectedConditions.elementToBeClickable(driver.findElement(By.id

("element id"))));

}

In the above method WebDriver would wait for the expected condition

(elementToBeClickable) to be met or for the timeout (20 seconds) to occur. As soon

as the condition is met, the WebDriver would execute the next test step

78.In XPath, I wants to do partial match on attribute value from beginning. Tell me two functions using which I can do It.

We can use bellow given two functions with XPath to find element using attribute value from beginning.

contains()

starts-with()

Where to use these:

– When we do not have complete knowledge about the web elements HTML properties

– When the values of the attributes are dynamic i.e. changing

– When we would like to create a list of web elements containing same partial attribute value

BY Attributes:

driver.findElement(By.xpath("//input[contains(@name,'user_name')]")).sendKeys("ad min");

By Text():

driver.findElement(By.xpath("//a[contains(text(), 'Marketing')]")).click();

Market ing (index.php?module=Campaigns&action=index&parenttab=Marketing)

Starts-with()

starts-with() method is used when we know about the initial partial attribute value or initial partial text associated with the web element. User can also use this method to locate web elements those are consist of both the static(initial) and dynamic(trailing) values.

By Attribute – //a[starts-with(@id,'link-si')]

– //a[starts-with(@id,'link-sign')]

79.How to know the selected box is already selected using selenium?

By using isSelected() to check that check box is already selected or not.

if(!(driver.findElement(By.id(Objects.getProperty(object))).isSelected())){
driver.findElement(By.id(Objects.getProperty(object))).click(); System.out.println("Checkbox "+object+ " has been selected");

80. How to verify the text inside the text box is present or not?

String Textboxvalue = “”;

Textboxvalue=driver.findElement(By.xpath(Objects.getProperty(object))).getAttribut e("value");

if(Textboxvalue.contains(expectedData)){ message="Pass";

81.How to accept exceptions in testNG?

using the @Test(expectedExceptions = ArithmeticException.class, NullPointerException.class)

82.I have used findElements In my test case. It Is returning NoSuchElementException when not element found. Correct me If I am wrong

It Is Incorrect. findElements will never return NoSuchElementException. It will return just an empty list.

83.My Firefox browser Is not Installed at usual place. How can I tell FirefoxDriver to use It?

If Firefox browsers Is Installed at some different place than the usual place then you needs to provide the actual path of Firefox.exe file as bellow.

System.setProperty("webdriver.firefox.bin","C:\Program Files\Mozilla Firefox\Firefox.exe");

driver =new FirefoxDriver();

84.How to handle Untrusted SSL certificate error in IE browser

System.setProperty("webdriver.ie.driver", "path of IEDriverServer.exe ");

DesiredCapabilities capabilities = DesiredCapabilities.internetExplorer();

// this line of code is to resolve protected mode issue

capabilities.setCapability(InternetExplorerDriver.INTRODUCE_FLAKINESS_BY_IGNORIN G_SECURITY_DOMAINS, true);

capabilities.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);

driver = new InternetExplorerDriver(capabilities);

85.Can we run testNG class code without using any TestNg annotation?

No, you have to add one “@Test” annotation compulsory

86.What Is TestNG?

TestNG Is Open Source(Freeware) framework which Is Inspired from NUnit and JUnit with Introducing few new features and functionality compared to NUnit and JUnit to make It easy to use and more powerful.

We can use TestNg with selenium webdriver to configure and run test cases very easily, easy to understand, read and manage test cases, and to generate HTML or XSLT test reports.

87.Can you describe major features of TestNG?

TestNG has many major features like support of @DataProvider annotation to perform data driven testing, can set test case execution dependency, test case grouping, generate HTML and XSLT test execution report etc..

88.How to Install TestNG In Eclipse? How do you verify that TestNg Is Installed properly In Eclipse?

To Install TestNG In Eclipse. We can directly install the TestNG in the Eclipse using help->eclipseMarket. Or we can download the .jar .

89.What are different annotations supported by TestNG ?

TestNG supports many different annotations to configure Selenium WebDriver test.

@Test

@Test annotation describes method as a test method or part of your test.

@Test (Priority = 1)

@ to the priority of the function

@Test(group = {“smoke”})

@ set the method name with group, when testNG will run “smoke” group only those

methods will run which are only belongs to the “smoke” group

@Parameters

When you wants to pass parameters in your test methods, you need to

use @Parameters annotation.

In the .xml file

(

In the class, use just above the methods

@Test

@Parameters ({"browser"})

@BeforeMethod

Any method which is marked with @BeforeMethod annotation will be executed before each and every @test annotated method.

@AfterMethod

Same as @BeforeMethod, If any method is annotated with @AfterMethod annotation then it will be executed after execution of each and every @test annotated method.

@BeforeClass

Method annotated using @BeforeClass will be executed before first @Test method execution. @BeforeClass annotated method will be executed once only per class so don't be confused.

@AfterClass

Same as @BeforeClass, Method annotated with @AfterClass annotation will be executed once only per class after execution of all @Test annotated methods of that class.

@BeforeTest

@BeforeTest annotated method will be executed before the any @Test annotated method of those classes which are inside tag in testng.xml file.

@AfterTest

@AfterTest annotated method will be executed when all @Test annotated methods completes its execution of those classes which are inside tag in testng.xml file.

@BeforeSuite

Method marked with @BeforeSuite annotation will run before the all suites from test.

@AfterSuite

@AfterSuite annotated method will start running when execution of all tests executed from current test suite.

@DataProvider

When you use @DataProvider annotation for any method that means you are using that method as a data supplier. Configuration of @DataProvider annotated method must be like it always return Object[][] which we can use in @Test annotated method.

If you want to provide the test data, the DataProvider way, then we need to declare a method that returns the data set in the form of two dimensional object array Object[][]. The first array represents a data set whereas the second array contains the parameter values.

There are two way to provide data using @DataProvideer annotation Dynamic way: passing data through another method:

We have provided the DataProvider method getData within the test class itself. Note that it is annotated with @DataProvider. Since it doesn’t have the name attribute, its name by default will be getData. It returns two sets of data, each set of which contains two values, an integer and a string value.

Public class InstanceDataProviderExample {

@Test(dataProvider="getData")

public void instanceDbProvider(int p1, String p2) {

System.out.println("Instance DataProvider Example: Data(" + p1 + ", " + p2 + ")");

}

@DataProvider

public Object[][] getData() {

return new Object[][]{{5, "five"}, {6, "six"}};

}

}

OUTPUT:

[TestNG] Running:

C:\javacodegeeks_ws\testNgDataProvider\test\com\javacodegeeks\testng\testng.xml

Instance DataProvider Example: Data(5, five)

Instance DataProvider Example: Data(6, six)

Static Data Provider:DataProvider method can also be defined in a separate class as a static method, in which case, the test method using it has to specify both the DataProvider name and its class in the @Test attributes dataProvider and dataProviderClass.

public class StaticDataProviderExample {

@Test(dataProvider="client1", dataProviderClass=DataProviderSource.class)

public void client1Test(Integer p) {

System.out.println("Client1 testing: Data(" + p + ")");

}

@Test(dataProvider="client2", dataProviderClass=DataProviderSource.class)

public void client2Test(Integer p) {

System.out.println("Client2 testing: Data(" + p + ")");

}

}

OUTPUT:

C:\javacodegeeks_ws\testNgDataProvider\test\com\javacodegeeks\testng\staticDataPr oviderTestng.xml

Client1 testing: Data(1)

Client2 testing: Data(2)

@BeforeGroups

@BeforeGroups annotated method will run before the first test run of that specific group.

@AfterGroups

@AfterGroups annotated method will run after all test methods of that group completes its execution.

@Factory

When you wants to execute specific group of test cases with different values, you need to use @Factory annotation. An array of class objects is returned by @Factory annotated method and those TestNG will those objects as test classes.

@Listeners

90.What Is the usage of testng.xml file?

In selenium WebDriver, We are using testng.xml file to configure our whole test suite In single file. Few of the tasks which we can specify In testng.xml file are as below. We can define test suite using set of test cases to run them from single place.

Can Include or exclude test methods from test execution.

Can specify a group to Include or exclude.

Can pass parameter to use In test case.

Can specify group dependencies.

Can configure parallel test execution.

Can define listeners.

91.How to pass parameter with testng.xml file to use It In test case?

We can define parameter In testng.xml file using syntax like bellow.

Here, name attribute defines parameter name and value defines value of that parameter. Then we can use that parameter In selenium webdriver test case using bellow given syntax.

@Parameters ({"browser"})

92.I have a test case with two @Test methods. I want to exclude one @Test method from execution. Can I do It? How?

```
Yes you need to specify @Test method exclusion In testng.xml file as bellow.

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name="Test Exclusion Suite">

<test name="Exclusion Test" >

<classes>

<class name="Your Test Class Name">

<methods>

<exclude name="Your Test Method Name To Exclude"/>

</methods>

</class>

</classes>
```

```
</test>

</suite>

You need to provide @Test method name In exclude tag to exclude It from execution.
```

93.Tell me syntax to skip @Test method from execution.

You can use bellow given syntax Inside @Test method to skip It from test execution. Please see the code in the “package framework.vTigerDay1.TestNGClass and TestNG.xml file;”

Using (@Test (enabled= false)): we can skip any method even it include in the TestNG.xml file throw new SkipException("Test Check_Checkbox Is Skipped");

```
# 94.Arrange bellow give testng.xml tags from parent to child.

<test>

<suite>

<class>

</methods>

</classes>
```

```
Parent to child arrangement for above testng tags Is as bellow.

<suite>

<test>

</classes>

<class>

</methods>
```

94.Tell me any 5 assertions of TestNG which we can use In selenium webdriver.

There are many different assertions available In TestNG but generally I am using bellow given assertions In my test cases.

assertEquals: This assertion is useful to compare expected and actual values in selenium webdriver. If both values match then its fine and will continue execution. But if fails then immediately it will mark that specific test method as fail and exit from that test method.

Assert.assertEquals(actual, expected);

Assert.assertEquals(Actualtext, "Tuesday, 28 January 2014")

assertNotEquals :It's function is opposite to assertEquals assertion. Means if both sides values will not match then this assertion will pass else it will fail Assert.assertNotEquals(actual, expected, message)

Assert.assertNotEquals(Actualtext, "Tuesday, 28 January 2014", "Expected and actual match in assertion_method_1");

assertTrue : assertTrue assertion is generally used for boolean condition true. It will pass if condition returns "true". If it will return false then it will fail and skip test execution from that specific method.

assertTrue(condition)

Assert.assertTrue(driver.findelement(By.id(“ ”)).isSelected());

assertFalse :It will check boolean value returned by condition and will pass if returned value is "False". If returned value is pass then this assertion will fail and skip execution from current test method.

Assert.assertFalse(condition)

Assert.assertFalse(driver.findelement(By.id(“ ”)).isSelected());

assertNull : assertNull(object) assertion will check and verify that object is null. It will pass if object found null. If object found not null then it will return error message like "java.lang.AssertionError: expected [null] but found [true]". Whenever your assertion fails, it will mark that specific test method as fail in TestNG result.

assertNull(object)

txt1 = driver.findElement(By.xpath("//input[@id='text1']"));

Assert.assertNull(txt1.getAttribute("disabled"));

assertNotNull: assertNotNull assertion is designed to check and verify that values returned by object is not null. Means it will be pass if returned value is not null. assertNotNull(object)

txt1 = driver.findElement(By.xpath("//input[@id='text1']"));

Assert.assertNotNull(txt1.getAttribute("disabled"));

96.Can you tell me usage of TestNG Soft Assertion?

Using TestNG soft assertion, We can continue our test execution even if assertion fails. That means on failure of soft assertion, remaining part of @Test method will be executed and assertion failure will be reportedat the end of @Test method.

To use testng soft assertion, you have to use testng SoftAssert class. This class will helps to not throw an exception on assertion failure and recording failure. If you will use soft assertion then your test execution will remain continue even If any assertion fails. Another most Important thing Is your assertion failure will be reported In report so that you can view It at end of test. You can use soft assertion when you are using multiple assertions In same te st method and you wants to execute all of them even If any one In between fails.

//Created object of testng SoftAssert class to use It's Properties.

SoftAssert s_assert = new SoftAssert();

//Text on expected side Is written Incorrect intentionally to get fail this assertion.

Assert.assertEquals(Actualtext, "Tuesday, 01 January 2014", "1st assert failed.");

System.out.println("Hard Assertion -> 1st pagetext assertion executed."

//Text on expected side Is written Incorrect intentionally to get fail this assertion.

s_assert.assertEquals(Actualtext, "Tuesday, 01 January 2014", "1st assert failed.");

System.out.println("Soft Assertion -> 1st pagetext assertion executed.");

97.How to write regular expression In testng.xml file to search @Test methods containing "product" keyword.

```
Regular expression to find @Test methods containing keyword "product" Is as bellow.

<methods>

<include name=".*product.*"/>

</methods>
```

98.Which time unit we provide In time test? minutes? seconds? milliseconds? or hours? Give Example

Time unit we provide on @Test method level or test suite level Is In milliseconds.

99.What Is Parallelism In TestNG?

```
In general software term, Parallelism means executing two part of
program simultaneously or executing program simultaneously or we can
say multithreaded or parallel mode. TestNG has same feature using which we can
start multiple threads simultaneously In parallel mode and test methods will be
executed In them.

Example:

<suite name="Parallel Class Suite" parallel="classes" thread-count="2">

<test name="Parallel Class Test" >

<classes>

<class name="Testing_Pack.Test_One"/>

<class name="Testing_Pack.Test_Two"/>

</classes>

</test>

</suite>
```

100.What Is dependency test In TestNG?

Dependency Is very good feature of testng using which we can set test method as dependent test method of any other single or multiple or group of test methods. That means depends-on method will be executed first and then dependent test method will be executed. If depends-on test method will fail then execution of dependent test method will be skipped automatically. TestNG dependency feature will works only If depends-on test method Is part of same class or part of Inherited base class.

Exmaple:

@Test(dependsOnMethods={"Login","checkMail"})

public void LogOut() {

System.out.println("LogOut Test code.");

}

101.How to use grid and while using grid how to run webDriver on ode machines?

```
To use grid we need two or multiple machines.
One would be host and other would be node.
HOST Machine:
Open command prompt and fire the below command or save that command in the
.bat file. So that we can run by using file it self.
D:
cd D:\seleium tool\Guru 99\WebDriver Test cases\SeleniumClassMy\jars
java -jar selenium-server-standalone-2.45.0.jar -role hub
Node Machine:
Open command prompt and file the below command or save that command in the
.bat file. So that we can run by using file itself.
And for using node machine we have to set the limitations and capabilities for the
node machine so that save all the configurations, limitations and capabilities in the
.json file. As all the files are in that path “D:\seleium tool\Guru 99\WebDriver Test
cases\SeleniumClassMy\grid”
D:
cd D:\seleium tool\Guru 99\WebDriver Test cases\SeleniumClassMy
java -jar jars\selenium-server-standalone-2.45.0.jar -role node -hub
http://localhost:4444/grid/register nodeconfig grid\nodeConfig.json
In the code, how to set the webDriver for grid?
publicstatic LoginPage openWebPageGrid(String browserName, String url, String OS)
throws MalformedURLException {
DesiredCapabilities dc = null;
if (browserName.equalsIgnoreCase("FF") &&OS.contentEquals("windows")){
dc = DesiredCapabilities.firefox();
dc.setPlatform(Platform.WINDOWS);
}
elseif (browserName.equalsIgnoreCase("CH") &&OS.equalsIgnoreCase("windows")){
System.setProperty("webdriver.chrome.driver", "Drivers\\chromedriver.exe");
dc = DesiredCapabilities.chrome();
dc.setPlatform(Platform.WINDOWS);
}
elseif (browserName.equalsIgnoreCase("IE") &&OS.equalsIgnoreCase("Windows")) {
System.setProperty("webdriver.ie.driver", "Drivers\\IEDriverServer.exe");
dc = DesiredCapabilities.internetExplorer();
dc.setPlatform(Platform.WINDOWS);
} // same we can create more conditions
// WD is the predefine key word, wd- webdriver
URL urlObj = newURL("http://14.98.122.21:4444/wd/hub");
driver = new RemoteWebDriver(urlObj, dc);
driver.get(url);
driver.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);
driver.manage().window().maximize();
return PageFactory.initElements(driver, LoginPage.class);
}
```

102.Can we use implicitly wait() and explicitly wait() together in the test case?

No. we should not..

Before using the explicitly wait(), reset the implicitly wait() by using the below code.

Create a method for the explicitly wait():

Public void explicitlyWaitForWebElement(WebDriver driver, WebElement we){

driver.manage.timeouts().implicitywait(0, TimeUnit.SECONDS); // nullify the time

WebDriverWait wait = new WebDriverWait(driver, 10);

Wait.until(ExpectedConditions.presenceOfElementLocated(we));

driver.manage().timeouts().implicitywait(DEFAULT_WAIT_4_PAGE,

TimeUnit.SECONDS); // reset the time

} Always call this function to set the explicitly wait().

103.What is the syntax of URL?

Uniform Resource Locator

resource_type://hostname.domain:port/path/filename

 resource_type - defines the type of Internet service (most common is http, ftp,https)

 HTTP: hyper text transport protocol

 HTTPS: Secure Hyper Text transport protocol

 hostname - defines the domain host (default host for http is www)

 domain - defines the Internet domain name (w3schools.com)

 port - defines the port number at the host (default for http is 80)

 path - defines a path at the server (If omitted: the root directory of the site)

 filename - defines the name of a document or resource

104.What is cookies?

Cookies are text files retained on computers by browsers containing various information in regards to a specific website visit.

Cookies are used to identify users, store shopping cart information, and prepare customized web pages that utilize user information. The cookie may be used to remember a username, help you resume where you left off, for example, so that the name will auto-fill on the user's next visit. Cookies may be disabled, or cookie options customized, due to privacy concerns and the risk of some cookies being used as spyware. It should be noted that because cookies are not executable files, they cannot be considered viruses as they do not have the ability to replicate.

Session cookies last only for as long as a user is on a website; they expire after the browser window is closed or the session times out.

Persistent cookies (also known as tracking cookies) remain active for a period of time on a user's machine and are used whenever the website is accessed. Secure cookies are used when accessing a website via HTTPS and are encrypted for greater safety.

Cookies cannot carry viruses, and cannot install malware on the host computer

105.What types of Models choose while designing the framework?

Most commonly use models are:

1. Behavioral Driven Development
2. Page Object Model
3. Behavioral Driven Development:
4. Page Object Model: In this type of Model, Each UI has different type of object (elements) to interact. These object are identified and written in the code along with their identity by using the One page have one Class in java.

Exp: @FindBy (name = “user_name”)

Public WebElement UserName;

@FindBy (xpath = “//span[@text()=’Marketing’]”)

Public WebElement Link;

Advantages: In page Object Model, one page objects save in the one class. This reduces the amount of duplicate code and If any UI changes occurs then need to change only in one place.

106. What is the frame? Why we need to switch to the frame?

```
frames are used to divide your browser window into multiple sections where each
section can load a separate HTML document.To use frames on a page we use
<frameset> tag instead of <body> tag. The <frameset> tag defines how to divide the
window into frames.


Frames are not the part of the same window handles because An IFrame (Inline
Frame) is an HTML document embedded inside another HTML document on a website.




IFrame:An inline frame is used to embed another document within the current HTML
document. It means iframe is actually a webpage within the webpage which have its
own DOM (Document object model ) for every iframe on the page. Web designers use
IFrames to embed interactive applications in Web pages, including those that employ
Ajax (Asynchronous JavaScript and XML), like Google Maps or ecommerce applications
```

107. What are the selenium exception you faced?

There are multiple exceptions in selenium:

NoSuchElementException:Thrown when element could not be found.

To handle this, use wait either implicit or explicit and first use WebElement.isDisplayed(), or WebElement.isEnable().

NoAlertPresentException:Thrown when switching to no presented alert.

NoSuchWindowException:Thrown when window target to be switched doesn’t exist.

NoSuchFrameException:Thrown when frame target to be switched doesn’t exist
ElementNotVisibleException:Thrown when an element is present on the DOM, but it is not visible, and so is not able to be interacted with.

To handle this exception, use explicit wait or try to find out the element by javaScript.

StaleElementReferenceException: This tells that element is no longer appearing on the DOM page.

To handle this exception use an explicit wait on the element to ensure the update is complete, then grab a fresh reference to the element again.

ElementNotSelectableException:Thrown when trying to select an unselectable element.

108.How to use @DataProvider annotation in TestNG?

@DataProvider When you use @DataProvider annotation for any method that means you are using that method as a data supplier. Configuration of @DataProvider annotated method must be like it always return Object[][] which we can use in @Test annotated method.

If you want to provide the test data, the DataProvider way, then we need to declare a method that returns the data set in the form of two dimensional object array Object[][]. The first array represents a data set whereas the second array contains the parameter values.

There are two way to provide data using @DataProvideer annotation Dynamic way: passing data through another method:

We have provided the DataProvider method getData within the test class itself. Note that it is annotated with @DataProvider. Since it doesn't have the name attribute, its name by default will be getData. It returns two sets of data, each set of which contains two values, an integer and a string value.

Public class InstanceDataProviderExample {

```
    @Test(dataProvider="getData")
```

public void instanceDbProvider(int p1, String p2) {

System.out.println("Instance DataProvider Example: Data(" + p1 + ", " + p2 +

")");

}

@DataProvider

public Object[][] getData() {

return new Object[][]{{5, "five"}, {6, "six"}};

}

}

OUTPUT:

[TestNG] Running:

C:\javacodegeeks_ws\testNgDataProvider\test\com\javacodegeeks\testng\testng.xml

Instance DataProvider Example: Data(5, five)

Instance DataProvider Example: Data(6, six)

Static Data Provider:DataProvider method can also be defined in a separate class as

a static method, in which case, the test method using it has to specify both the

DataProvider name and its class in the @Test attributes dataProvider and

dataProviderClass.

public class StaticDataProviderExample {

@Test(dataProvider="client1", dataProviderClass=DataProviderSource.class)

public void client1Test(Integer p) {

System.out.println("Client1 testing: Data(" + p + ")");

}

@Test(dataProvider="client2", dataProviderClass=DataProviderSource.class)

public void client2Test(Integer p) {

System.out.println("Client2 testing: Data(" + p + ")");

}

}

OUTPUT:

C:\javacodegeeks_ws\testNgDataProvider\test\com\javacodegeeks\testng\staticDataProviderTestng.xml

109.How to find out the dynamic element by webdriver?

Can find dynamic element by multiple ways:

 Absolute xpath:Xpath Position or Absolute Xpath are most frequently used to resolve the dynamic element issues.

web_element_name=html/body/div[30]/div[2]/div[2]/div/div/div/div[1]/table/tbody /tr/td[2]/table/tbody/tr/td[1]/table/tbody/tr/td[1]/table/tbody/tr[2]/td[2]/em/bu tton//p[6]/label[2]/div/ins

 Identity element by starts-with and contains text():

If the dynamic elements have a definite pattern to them, then we can also use JavaScript functions like “starts-with” or “contains” in our element locators to separate the dynamic part of locator from static part.

XPath: //button[starts-with(@id, 'Submit-')] XPath: //input[contains(@class, 'suggest')].

110.What are the most common pre define functions of xpath?

```
Below are the most common pre define functions are;
 Last()
 Position()
 Contains()
 Start-with ()
How to use them?
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book>
<title lang="en">Harry Potter</title>
<price>29.99</price>
</book>
<book>
<title lang="en">Learning XML</title>
<price>39.95</price>
</book>
</bookstore>
/ = Selects from the root node
```

```
// =Selects nodes in the document from the current node that match the selection
no matter where they are
. = Selects the current node
.. = Selects the parent of the current node
@ = Selects attributes
/bookstore = selects the root element bookstore
Note: If the path start with a slash(/), it always represents an absolute path to an
elements.
//book = select all book elements no matter where they are in the document
bookstore/book = Selects all book elements that are children of bookstore
bookstore//book = Selects all book elements that are descendant of the bookstore
element, no matter where they are under the bookstore element
//@lang = Selects all attributes that are named lang
//bookstore/book[1] =Selects the first book element that is the child of the bookstore
element.
Note: In IE 5,6,7,8,9 first node is[0], but according to W3C, it is [1]. To solve this
problem in IE, set the SelectionLanguage to XPath:
In JavaScript: xml.setProperty("SelectionLanguage","XPath");
//bookstore/book[last()]=Selects the last book element that is the child of the
bookstore element.
//bookstore/book[position()<3] = Selects the first two book elements that are
children of the bookstore element.
*= matches any elements node
@* = matches any attribute node
Node() = matches any node of any kind
/bookstore/* = select all the child element of the bookstore element.
//* = select all elements in the document.
//tittle[@*] = select all title elements which have at least one attribute of any
kind.
```

111. How to set the browser for the GRID?

```
For the grid we also need to add OS information and for the remote machine we use
RemoteWebDriver()


publicstatic LoginPage openWebPageGrid(String browserName, String url, String OS)
throws MalformedURLException {
DesiredCapabilities dc = null;
if (browserName.equalsIgnoreCase("FF") &&OS.contentEquals("windows")){
dc = DesiredCapabilities.firefox();
dc.setPlatform(Platform.WINDOWS);
}
elseif (browserName.equalsIgnoreCase("CH")
&&OS.equalsIgnoreCase("windows")){
System.setProperty("webdriver.chrome.driver",
"Drivers\\chromedriver.exe");
dc = DesiredCapabilities.chrome();
dc.setPlatform(Platform.WINDOWS);
}
elseif (browserName.equalsIgnoreCase("IE")
&&OS.equalsIgnoreCase("Windows")) {
System.setProperty("webdriver.ie.driver", "Drivers\\IEDriverServer.exe");
dc = DesiredCapabilities.internetExplorer();
dc.setPlatform(Platform.WINDOWS);
} // same we can create more conditions
// WD is the predefine key word, wd- webdriver
URL urlObj = newURL("http://14.98.122.21:4444/wd/hub");
driver = new RemoteWebDriver(urlObj, dc);
driver.get(url);
driver.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);
driver.manage().window().maximize();
return PageFactory.initElements(driver, LoginPage.class);
}
```

112.How to get the snapshot using selenium?

```
publicstaticvoid fn_takeSanpShot(String path) throws IOException{
if (fileSnapShotObj == null) {
fileSnapShotObj = new File(snapShotFolderPath);
if (!fileSnapShotObj.exists()){
// this is using to create the directory for snapShot if not exist.
fileSnapShotObj.mkdir();
}
}
TakesScreenshot tss = (TakesScreenshot)driver;
File scrObj = tss.getScreenshotAs(OutputType.FILE);
File destObj = newFile(path);
FileUtils.copyFile(scrObj, destObj);
 }
```

113. How to shoot the snapshot using selenium?

```
publicstaticvoid shootSnapShot(WebElement we, String destPath) {
if (fileSnapShotObj == null) {
fileSnapShotObj = new File(snapShotFolderPath);
if (!fileSnapShotObj.exists()){
// this is using to create the directory for snapShot if not exist.
fileSnapShotObj.mkdir();
}
}
TakesScreenshot tss = (TakesScreenshot) driver;
File srcObj = tss.getScreenshotAs(OutputType.FILE);
Point p = we.getLocation();
intheight = we.getSize().getHeight();
intwidth = we.getSize().getWidth();
System.out.println("p.getx()= "+p.getX()+"p.gety()="+p.getY()+"height=
"+height+"width"+width);
try {
BufferedImage biObj = ImageIO.read(srcObj);
// Parameters:
//x - the X coordinate of the upper-left corner of the specified rectangular region
//y - the Y coordinate of the upper-left corner of the specified rectangular region
//w - the width of the specified rectangular region
//h - the height of the specified rectangular region
BufferedImage dest = biObj.getSubimage(p.getX(), p.getY(), width, height);
File destObj = newFile(destPath);
ImageIO.write(dest, "jpeg", srcObj);
FileUtils.copyFile(srcObj, destObj);
Reporter.log("<a href=\"" + destPath + "\"> SnapShot </a>" );
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
 }
}
```

114.How to switch on the given URL in multiple windows handle?

```
First find all the windows handle and then save it in the Iterator and then check the
url of every windows and then switch.
publicstaticvoid fn_switchWindowUsingURL(String fieldName){
String fieldValue = fn_getValue(fieldName);
```

```
Set<String>setObj = driver.getWindowHandles();
Iterator<String>iteObj = setObj.iterator();
while(iteObj.hasNext()) {
driver.switchTo().window(iteObj.next());
String currentURL = driver.getCurrentUrl();
if(currentURL.contains(fieldValue)){
System.out.println("Contains the url");
break;
}
}
}
```

115.Explain the different between HTTP and HTTPS?

The differences between HTTP and HTTPS are following:

 Hypertext Transfer Protocol is a protocol for information to be passed back and forth between web servers and clients. Https is refers to the combination of a normal HTTP interaction over an encrypted Secure Sockets Layer (SSL) or Transport Layer Security (TLS) transport mechanism

 HTTP use port number 80 whereas HTTPS use port number 443

 HTTP can support the client asking for a particular file to be sent only if it has been updated after a certain date and time whereas Hypertext Transfer Protocol over Secure Socket Layer is built into its browser that encrypts and decrypts user page requests as well as the pages that are returned by the Web serve

116. How do you read data from excel ?

FileInputStream fis = new FileInputStream(“path of excel file”);

Workbook wb = WorkbookFactory.create(fis);

Sheet s = wb.getSheet(“sheetName”);

String value = s.getRow(rowNum).getCell(cellNum).getStringCellValue();

117. What is the alternate way to click on login button?

use submit() method but it can be used only when attribute type=submit.

118.How do you verify if the checkbox/radio is checked or not ?

We can use isSelected() method.

Syntax –

driver.findElement(By.xpath("xpath of the checkbox/radio button")).isSelected();

If the return value of this method is true then it is checked else it is not

119.How do you handle alert pop-up ?

To handle alert pop-ups, we need to 1st switch control to alert pop-ups then click on ok or cancle then move control back to main page. .

Syntax□String mainPage = driver.getWindowHandle();

Alert alt = driver.switchTo().alert(); → to move control to alert popup

alt.accept(); ---> to click on ok.

alt.dismiss(); ---> to click on cancle.

Then move the control back to main web page□driver.switchTo().window(mainPage); → to switch back to main page.

120.Give the example for method overload in WebDriver.

frame(string), frame(int), frame(WebElement).

121.How do you upload a file?

To upload a file we can use sendKeys() method. Syntax - driver.findElement(By.xpath("input field")).sendKeys("path of the file which u want to upload");

122.How do you click on a menu item in a drop down menu?

if that menu has been created by using select tag then we can use the methods selectByValue() or selectByIndex() or selectByVisibleText(). These are the methods of the Select class.

If the menu has not been created by using the select tag then we can simply find the xpath of that element and click on that to select.

123.How do you clear the contents of a textbox in selenium ?

use clear() method.

syntax- driver.findElement(By.xpath("xpath of box")).clear();

124. How to get the number of frames on a page ?

List framesList = driver.findElements(By.xpath("//iframe"));

int numOfFrames = frameList.size();

125. How do you simulate scroll down action ?

use java script to scroll down-

JavascriptExecutor jsx = (JavascriptExecutor)driver;

jsx.executeScript("window.scrollBy(0,4500)", ""); //scroll down, value 4500 you

can change as per your req

jsx.executeScript("window.scrollBy(450,0)", ""); //scroll up

ex□public class ScrollDown {

public static void main(String[] args) throws InterruptedException {

WebDriver driver = new FirefoxDriver();

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

driver.get("http://www.flipkart.com/womens□clothing/pr?sid=2oq,c1r&otracker=hp_nmenu_sub_women_1_View%20all" (http://www.flipkart.com/womens%02clothing/pr?sid=2oq,c1r&otracker=hp_nmenu_sub_women_1_View%20all"));

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

JavascriptExecutor jsx = (JavascriptExecutor)driver;

jsx.executeScript("window.scrollBy(0,4500)", ""); //scroll down

Thread.sleep(3000);

jsx.executeScript("window.scrollBy(450,0)", ""); //scroll up

}

}

126. What is the command line we have to write inside a .bat file to execute a selenium project when we are using testng ?

java -cp bin;jars/* org.testng.TestNG testng.xml

127.How to check if a text is highlighted on the page ?

```
To identify weather color for a field is different or not-
 String color =
driver.findElement(By.xpath("//a[text()='Shop']")).getCssValue("color");
 String backcolor =
driver.findElement(By.xpath("//a[text()='Shop']")).getCssValue("background-color");
 System.out.println(color);
 System.out.println(backcolor);
 Here if both color and backcolor different then that means that element is in
```

```
different color.
```

128.How do u get the width of the textbox ?

driver.findElement(By.xpath("xpath of textbox ")).getSize().getWidth();

driver.findElement(By.xpath("xpath of textbox ")).getSize().getHeight();

129. How to check whether a text is underlined or not ?

```
Identify by getCssValue("border-bottom") or sometime getCssValue("text•decoration")
method if the
 cssValue is 'underline' for that WebElement or not.
 ex- This is for when moving cursor over element that is going to be underlined or
not-
 public class UnderLine {
 public static void main(String[] args) {
 WebDriver driver = new FirefoxDriver();
 driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
 driver.get("https://www.google.co.in/?gfe_rd=ctrl&ei=bXAwU8jYN4W6iAf8zIDgD
A&gws_rd=cr");
String cssValue=
driver.findElement(By.xpath("//a[text()='Hindi']")).getCssValue("text-decoration");
 System.out.println("value"+cssValue);
 Actions act = new Actions(driver);
 act.moveToElement(driver.findElement(By.xpath("//a[text()='Hindi']"))).perform
();
 String cssValue1=
driver.findElement(By.xpath("//a[text()='Hindi']")).getCssValue("text-decoration");
 System.out.println("value over"+cssValue1);
 driver.close();
 }
}
```

130.What is the use of getOptions() method ?

getOptions() is used to get the selected option from the dropdown list.

131.What is the use of deSelectAll() method ?

It is used to deselect all the options which have been selected from the dropdown list

132.Which is the super interface of webdriver ?

SearchContext

133. How to enter text without using sendkeys()

```
Yes we can enter text without using sendKeys() method. We have to use combination
of javascript and wrapper classes with WebDriver extension class, check the below
code•public static void setAttribute(WebElement element, String
```

```
attributeName, String value)
{
WrapsDriver wrappedElement = (WrapsDriver) element;
JavascriptExecutor driver = (JavascriptExecutor)
wrappedElement.getWrappedDriver();
driver.executeScript("arguments[0].setAttribute(arguments[1],
arguments[2])", element, attributeName, value);
}
call the above method in the test script and pass the text field attribute and pass
the
text you want to enter.
```

134. There is a scenario whenever "Assert.assertEquals()" function fails automatically it has to take screenshot. How can you achieve this ?

By using EventFiringWebDriver.

Syntax□EventFiringWebDriver eDriver=new EventFiringWebDriver(driver);

File srcFile = eDriver.getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(srcFile, new File(imgPath))

135.How do you handle https website in selenium

```
By changing the setting of FirefoxProfile.
Syntax•public class HTTPSSecuredConnection {
 public static void main(String[] args){
 FirefoxProfile profile = new FirefoxProfile();
 profile.setAcceptUntrustedCertificates(false);
 WebDriver driver = new FirefoxDriver(profile);
 driver.get("url");
 }
}
```

136. How do you send ENTER/TAB keys in webdriver?

```
use click() or submit() [submit() can be used only when type='submit']) method for
ENTER.
Or act.sendKeys(Keys.ENTER);
For Tab-
 act.sendKeys(Keys.TAB)
    where act is Actions class type. ( Actions act = new Actions(act); )
```

137. While explaining the framework, what are points which should be covered ?

1.What is the frame work.

2. Which frame work you are using.
3. Why This Frame work.

4. Architecture.
5. Explanation of every component of frame work.
6. Process followed in frame work.
7. How & when u execute the frame work.
8. Code (u must write code and explain).
9. Result and reporting .
10. You should be able to explain it for 20 Minutes.

138. How to switch back from a frame ?

use method defaultContent().

Syntax – driver.switchTo().defaultContent();

139.How to type text in a new line inside a text area ?

Use \n for new line.

ex- webelement.sendKeys("Sanjay_Line1.\n Sanjay_Line2.");

it will type in text box as

Sanjay_Line1.

Sanjay_Line2.

140. What is the use of AutoIt tool ?

Some times while doing testing with selenium, we get stuck by some interruptions like a window based pop up. But selenium fails to handle this as it has support for only web based application. To overcome this problem we need to use AutoIT along with selenium script. AutoIT is a third party tool to handle window based applications. The scripting language used is in VBScript.

141.How to press Shift+Tab ?

String press = Keys.chord(Keys.SHIFT,Keys.TAB);

webelement.sendKeys(press);

142.What is the use of contextClick() ?

It is used to right click.

143.What is the difference b/w getWindowHandles() and getWindowHandle() ?

getWindowHandles()- is used to get the address of all the open browser and its return type is Iterator.

getWindowHandle() is used to get the address of the current browser where the control is and return type

144.How do you accommodate project specific methods in your framework ?

1st go through all the manual test cases and identify the steps which are repeating. Note down such steps and make them as methods and write into ProjectSpecificLibrary.

145.What are different components of your framework

Library- Assertion, ConfigLibrary, GenericLibrary, ProjectSpecificLibrary, Modules. Drivers folder, Jars folder, excel file.

146.Count the number of links in a page.

use the locator By.tagName and find the elements for the tag //a then use loop to count the number of elements found.

Syntax- int count = 0;

List link = driver.findElements(By.tagName("a"));

System.out.println(link.size()); // this will print the number of links in a pag

147. What is the difference between before method and before class ?

@BeforeMethod- this will execute before every @Test method.

@BeforeClass- this will execute before every class.

148.What are the different attributes for @Test annotation?

alwaysRun, dataProvider, dependsOnMethods, enabled, expectedExceptions, timeOut etc.

ex- @Test(expectedExceptions = ArithmeticException.class),

@Test(timeOut = 2000).

149.What is object repository ?

An object repository is a very essential entity in any UI automation tool. A repository allows a tester to store all the objects that will be used in the scripts in one or more centralized locations rather than letting them be scattered all over the test scripts. The concept of an object repository is not tied to WET alone. It can be used for any UI test automation. In fact, the original reason why the concept of object repositories were introduced was for a framework required by QTP.

150.What is testing strategy ?

A Test Strategy document is a high level document and normally developed by project manager. This document defines “Software Testing Approach” to achieve testing objectives. The Test Strategy is normally derived from the Business Requirement Specification document.

151.Difference between Web driver listener and TestNG Listener.

TestNG and Web driver Listener have different interfaces to implement and call them. They both modify respective behaviour. You can use Listeners in Annotation

152.Why we refer Firefox driver to the web driver inheritance.

```
web Driver driver = new FireFoxDriver();
WebDriver is an interface which contain several abstract methods such as get(...),
findElamentBy(...) etc.
We simply create reference of web Driver and we can assign objects (Firefox driver,
CromeDriver, IEDriver, Andriod driver etc) to it.
Ex :
WebDriver driver = new FireFoxDriver();-----------(1)
If we are using (1) we can do the same thing by using
FireFoxDriver driver = new FireFoxDriver();---------(2)
We can use (1) and (2) for same purpose but if we want to switch to another browser
in same program then again we have to create the object of other class as for
example
CromeDriver driver = new CromeDriver();.
creating object of several class is not good. So we create the reference of WebDriver
and we assign the objects of another class as for example
WebDriver driver; // it is created only one time in the program
driver = new FireFoxDriver();// any where in the program
driver = new CromeDriver(); // any where in the program
```

153.What is the difference between thread.Sleep() and selenium. Set Speed ("2000")?

```
If the application is taking time to load the page then we use
selenium.waitforpageload(" "). This command is doesn’t wait upto the given time
whenever the page load is completed.
If the application is taking time to refresh the page, then we use Thread. Sleep (
).it
is a standard wait it simply wait to the given time.
selenium.setSpeed
1. Takes a single argument in string format
Ex: selenium.setSpeed("2000") - will wait for 2 seconds
2. Runs each command in after setSpeed delay by the number of milliseconds
mentioned in set Speed.
thread.sleep
1. Takes a single argument in integer format
ex: thread. Sleep(2000) - will wait for 2 seconds
2. Waits for only once at the command given at sleep.
```

154.In what situation selenium finding element get fails?

 · Element loading issue · Dynamic id of web element

155. How we can retrieve the dynamically changing Ids?

It can be handled through customized xpath

 preceding-sibling

 following-sibling

 contains method

 starts-with() method

156.What is the basic use of Firefox profiles and how can we use them using selenium?

A profile in Firefox is a collection of bookmarks, browser settings, extensions, passwords, and history; in short, all of your personal settings. We use them to change user agent, changing default download directory, changing versions etc

157.Customize the name of file going to be downloaded?

```
You have to download AUTO IT.exe file and has to be install and later you have
create .au3 file (in this file you have to specify the commands in
VB script like your file name, where have to save, it will be easy may be 3 or 4
steps )
using AUTOIT...then right click the .au3 file you have to compile ....after that you
will
get the .exe file with the name of .au3 file ..In eclipse you will give the code like
this
<----ProcessBuildderps = new ProcessBuilder("path of the .exe file of au3")
.start();--->
```

158.How to handle internationalisation throughweb driver?

```
FirefoxProfile profile = new FirefoxProfile();
profile.set Preference("intl.accept_languages","jp");
Web driver driver = new FirefoxDriver(profile); driver.get(google.com) will open
google in
Japanese Lang
```

159. How to overcome same origin policy through web driver?

```
 · Proxy server.
DesiredCapabilities capability=new DesiredCapabilities.firefox();
capability.setCapability(CapabilityType.PROXY,"your desire proxy")
WebDriver driver=new FirefoxDriver(capability);
```

160. Difference between flex and flash application.

In flash there is no code just based on creativity(design) we will complete the work(time consuming process) whereas flex contain some small functions which is integrated with mxml,PHP..(no tool is there to developflex we want to use the properties of css and style sheet)

161.What is Error Collector in TestNG? What is its use?

This class allows the collection of errors during the process of retrieving the test data for the test method parameters

162. How to run tests in multiple browser parallel? Is there any other option other than selenium grid?

```
You create a class with a method something like this:
public class LaunchBrowser {
WebDriver driver=null;
// Pass parameter browser from test.xml
@Parameters(“browser”)
public void initiateBrowser(String browser){
// compare browser to fire fox and then open firefox driver
if(browser.equals(“Firefox”))
{
driver = new FirefoxDriver();
}
else
{
\ set path to the IE driver correctly here
System.setProperty("webdriver.ie.driver", "\iexploredriver.exe");
driver =new InternetExplorerDriver();
}
}
Now create YourClassName class and call extend the above class something like this
@Test
public class YourClassName extends LaunchBrowser{
public void gotoGoogle(){
driver.get(“http://www.google.com");
}
}
```

163.) How to prepare Customized html Report using TestNG in hybrid framework.

• Junit: with the help of ANT.

• TestNG: using inbuilt default.html to get the HTML report. Also XST reports from ANT,

Selenium, TestNG combination.

• Using our own customized reports using XSL jar for converting XML content to HTML.

164. How the TestNG interacts with Selenium Core? Explain me steps and internal architecture?"What is TestNG?

So far we had been doing Selenium tests without generating a proper format for the test results. From this point on, we shall tackle how to make these reports using a test framework called TestNG.

TestNG is a testing framework that overcomes the limitations of another popular testing framework called JUnit. The "NG" means "Next Generation". Most Selenium users use this more than JUnit because of its advantages. There are so many features of TestNG, but we will only focus on the most important ones that we can use in Selenium.Advantages of TestNG over JUnit

There are three major advantages of TestNG over JUnit:

 Annotations are easier to understand

 Test cases can be grouped more easily

 Parallel testing is possible

165. Is it possible test web services using selenium?

Using Jmeter we can test how one website is talking to each other means time taken to send data, feeds, messages from one website to other website. Jmeter does a nice job of doubling for performance and api tests

166.How to refresh a page without using context click

```
1.UsingsendKeys.Keys method

2.Usingnavigate.refresh() method

3.Usingnavigate.refresh() method

4.Using get() method

5.UsingsendKeys() method
1.UsingsendKeys.Keys method
driver.get("https://accounts.google.com/SignUp");
driver.findElement(By.id("firstname-placeholder")).sendKeys(Keys.F5);
2.Usingnavigate.refresh() method
```

```
driver.get("http://ruchi-myseleniumblog.blogspot.in/2013/12/100-selenium•interview-
questions.html");
driver.navigate().refresh();
3.Using navigate.to() method
driver.get("http://ruchi-myseleniumblog.blogspot.in/2014/01/selenium-hybrid•framework-
using.html");
driver.navigate().to(driver.getCurrentUrl());
4.Using get() method
driver.get("http://ruchi-myseleniumblog.blogspot.in/2013/12/basic-core-java•interview-
questions.html");
driver.get(driver.getCurrentUrl());
5.UsingsendKeys() method
driver.get("https://accounts.google.com/SignUp");
driver.findElement(By.id("firstname-placeholder")).sendKeys("\uE035");
```

167.Can you send a code for printing in selenium?

```
There are two cases:
Case1. Any hyperlink/button on a web page, n clicking that link/button a print dialog
box
opens. (Performing an action on web page)
Case2.or do u want to open print dialog box within ur own script, not by performing
any
action on web page.
So If Case 1: just a call for WebElement.click() event will work to open it.
If Case 2: Call a Printer Job object (Use Awt API).
For code: Google it.
```

168.How to find broken images in a page using Selenium Web driver.

```
1. Get xpath and then using tag name; get all the links in the page
2. Click on each and every link in the page
3. In the target page title, look for 404/500 error.
How to find broken images in a page using Selenium
package programs;
import java.util.List;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
public class findbrokenimages {
static intinvalidimg;
static WebDriver driver ;
public static void main(String[] args) {
try {
driver = new FirefoxDriver();
driver.get("http://ruchi-myseleniumblog.blogspot.in");
invalidimg = 0;
List allImages = driver.findElements(By.tagName("img"));
System.out.println("Total images are " + allImages.size());
for (int i = 0; i <allImages.size(); i++) {
WebElementimg = (WebElement) allImages.get(i);
if (img != null) {
verifyimgActive(img);
}
}
```

```
System.out.println("Total invalid images are " + invalidimg);
driver.quit();
} catch (Exception e) {
e.printStackTrace();
System.out.println(e.getMessage());
}
}
public static void verifyimgActive(WebElementimg) {
try {
HttpResponse response = new DefaultHttpClient().execute(new
HttpGet(img.getAttribute("src")));
if (response.getStatusLine().getStatusCode() != 200)
invalidimg++;
}
catch (Exception e) {
e.printStackTrace();
}
}
}
```

169. How to handle Ajax popup window?

By using getWindowHandles() and obj.switchTo.window(windowid) we can handle popups using explicit wait and driver.swtchT0.window("name") commands for your requirements.

170. How to handle auto complete box in web driver?

```
How to handle autocomplete box in web driver
How to handle autocomplete box in web driver?
driver.findElement(By.id("your searchBox")).sendKeys("your partial keyword");
Thread.sleep(3000);
List <WebElement>listItems = driver.findElements(By.xpath("your list item locator"));
listItems.get(0).click();
driver.findElement(By.id("your searchButton")).click();
```

171.How to get the name of browser using Web Driver?

```
public class JsExecute
{
WebDriver driver;
JavascriptExecutorjs;
@Before
public void setUp() throws Exception
{
driver=new FirefoxDriver();
driver.get("http://www.google.com");
}
@Test
public void test()
{
JavascriptExecutorjs = (JavascriptExecutor) driver;
System.out.println(js.executeScript("return navigator.appCodeName"));
}}
OR
String s = (String) ((JavascriptExecutor) driver).executeScript("return
navigator.userAgent;");
System.out.println("Browser name : " + s);
```

172.How to pass parameters from testng.xml into test case.

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.htmlunit.HtmlUnitDriver;
import org.openqa.selenium.ie.InternetExplorerDriver;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;
public class Parallelexecution {
private WebDriver driver = null;
@BeforeTest
@Parameters({ "BROWSER" })
public void setup(String BROWSER) {
System.out.println("Browser: " + BROWSER);
if (BROWSER.equals("FF")) {
System.out.println("Firefox Browser is selected");
driver = new FirefoxDriver();
} else if (BROWSER.equals("IE")) {
System.out.println("Internet Explorer Browser is selected");
driver = new InternetExplorerDriver();
} else if (BROWSER.equals("HU")) {
System.out.println("Html Unit Browser is selected");
driver = new HtmlUnitDriver();
} else if (BROWSER.equals("CH")) {
System.out.println("Google chrome Browser is selected");
driver = new ChromeDriver();
}
}
@Test
public void testParallel() throws Exception {
driver.get("http://ruchi-myseleniumblog.blogspot.in/2013/12/100-selenium•interview-
questions.html");
}
}
```

173. How to get text from captcha image??

driver.findElement(By.xpath(".//*[@id='SkipCaptcha']")).click();

String attr = ie.findElement(By.xpath(".//*[@id='SkipCaptcha']")).getAttribute("value");

System.out.println("The value of the attribute 'Name' is " + attr);

174. Is there a way to click hidden LINK in web driver?

String Block1 = driver.findElement(By.id("element ID")); JavascriptExecutor js1= (JavascriptExecutor)driver; js1.executeScript("$("+Block1+").css({'display':'block'});");

175.) How to disable cookies in browser

Using deleteAllVisibleCookies() in selenium

176.We have heard about frameworks well it can be broadly classified into these TDD, BDD and ATDD frameworks .What's the Difference?

TDD- Test Driven Development, Behaviour Driven Development & Acceptance TestDriven Development

Well, you could see the above Acronyms buzzing over all Automation folks. I was not sure on what it means and How it differs each other. How each methodology will benefit? and where exactly it will help in the Development Life cycle.

Finally, after some analysis I had found out the differences and posting it here. Readers are always welcomed to correct me if I am wrong. First lets list out what exactly each methodology does means

TDD – Test Driven Development

Its also called test-driven design, is a method of software development in which unit testing is repeatedly done on source code. Write your tests watch it fails and then refactor it. The concept is we write these tests to check if the code we wrote works fine. After each test, refactoring is done and then the same or a similar test is performed again. The process is iterated as many times as necessary until each unit is functionally working as expected. TDD was introduced first by XP. I believe I have explained enough in simple terms.

BDD – Behaviour Driven Development Behavior-driven development combines the general techniques and principles of TDD with ideas from domain-driven design DDD-Domain Driven Testing

BDD is similar in many ways to TDD except that the word "test" is replaced with the word "Behaviour". It's purpose is to help the the folks devising the system (i.e., the developer) identify appropriate tests to write– that is, tests that reflect the behavior desired by the stakeholders. BDD is usually done in very English-like language helps the Domain experts to understand the implementation rather than exposing the code level tests. Its defined in a GWT format, GIVEN WHEN & THEN.

177. How to change user agent in Firefox by selenium web driver.

FirefoxProfile profile = new FirefoxProfile(); profile.setPreference("general.useragent.override", "some UA string"); Web Driver driver = new FirefoxDriver(profile);

178. How to handle network latency using selenium?

Using driver.manage.pageLoadingtime for network latency

179.) How to work with radio button in web driver?

```
We can select the value from the drop down by using 3 methods.
selectByVisibleText - select by the text displayed in drop down
selectByIndex - select by index of option in drop down
selectByValue - select by value of option in drop down
<select id="44"><option value="1">xyz</option>
<option value="2">abc</option>
<option value="3">pqr</option>
</select>
WebElement e = driver.findElement(By.id("44"));
```

```
Select selectElement=new Select(e);
// both of the below statements will select first option in the weblist
selectElement.selectByVisibleText("xyz");
selectElement.selectByValue("1");
```

180. Detail about TestNG Test Output folder.

It is the directory where reports are generated. Every time tests run in a suite, TestNG creates index.html and other files in the output directory.

181. Can we run group of test cases using TestNG?

Test cases in group in Selenium using TestNG will be executed with the below options. If you want to execute the test cases based on one of the group like regression test or smoke test @Test(groups = {"regressiontest", "smoketest"})

182.In what all case we have to go for “JavaScript executor”.

Consider FB main page after you login. When u scrolls down, the updates get loaded. To handle this activity, there is no selenium command. So you can go for javascript to set the scroll down value like driver.executeScript("window.scrollBy(0,200)", "");

183. What is actions class in web driver?

```
Actions class with web Driver help is Sliding element, Resizing an Element, Drag &
Drop, hovering a mouse, especially in a case when dealing with mouse over menus.
Dragging & Dropping an Element:
importorg.openqa.selenium.By;
importorg.openqa.selenium.WebDriver;
importorg.openqa.selenium.WebElement;
importorg.openqa.selenium.firefox.FirefoxDriver;
importorg.openqa.selenium.interactions.Actions;
public class testDragandDrop {
public static void main(String[] args) throws InterruptedException {
WebDriver driver = new FirefoxDriver();
driver.get("http://jqueryui.com/resources/demos/droppable/default.html");
WebElementdraggable = driver.findElement(By.xpath("//*[@id='draggable']"));
WebElement droppable = driver.findElement(By.xpath("//*[@id='droppable']"));
Actions action = new Actions(driver);
action.dragAndDrop(draggable, droppable).perform();
}
}
Sliding an Element:
importorg.openqa.selenium.By;
importorg.openqa.selenium.WebDriver;
importorg.openqa.selenium.WebElement;
importorg.openqa.selenium.firefox.FirefoxDriver;
importorg.openqa.selenium.interactions.Actions;
public class testSlider {
/**
* @paramargs
* @throws InterruptedException
*/
public static void main(String[] args) throws InterruptedException {
WebDriver driver = new FirefoxDriver();
```

```
driver.get("http://jqueryui.com/resources/demos/slider/default.html");
WebElement slider = driver.findElement(By.xpath("//*[@id='slider']/a"));
Actions action = new Actions(driver);
Thread.sleep(3000);
action.dragAndDropBy(slider, 90, 0).perform();
}
}
Re-sizing an Element:
importorg.openqa.selenium.By;
importorg.openqa.selenium.WebDriver;
importorg.openqa.selenium.WebElement;
importorg.openqa.selenium.firefox.FirefoxDriver;
importorg.openqa.selenium.interactions.Actions;
public class testResizable {
public static void main(String[] args) throws InterruptedException {
WebDriver driver = new FirefoxDriver();
driver.get("http://jqueryui.com/resources/demos/resizable/default.html");
WebElement resize = driver.findElement(By.xpath("//*[@id='resizable']/div[3]"));
Actions action = new Actions(driver);
action.dragAndDropBy(resize, 400, 200).perform();
}
```

184.Differences between jxl and ApachePOI.

 jxl does not support XLSX files

 jxl exerts less load on memory as compared to ApachePOI

 jxl doesn't support rich text formatting while ApachePOI does.

 jxl has not been maintained properly while ApachePOI is more up to date.

 Sample code on Apache POI is easily available as compare to jxl.

185. How to ZIP files in Selenium with an Example?

```
// Sample Function to make zip of reports
public static void zip(String filepath){
try
{
File inputFolder=new File('Mention file path her");
File outputFolder=new File("Reports.zip");
ZipOutputStream out = new ZipOutputStream(new BufferedOutputStream(new
FileOutputStream(outputFolder)));
BufferedInputStream in = null;
byte[] data = new byte[1000];
String files[] = inputFolder.list();
for (int j=0; j<files.length; i++)
{
in = new BufferedInputStream(new FileInputStream
(inputFolder.getPath() + "/" + files[j]), 1000);
out.putNextEntry(new ZipEntry(files[j]));
inttotalcount;
while((totalcount= in.read(data,0,1000)) != -1)
{
out.write(data, 0, totalcount);
}
out.closeEntry();
}
out.flush();
out.close();
```

```
}
catch(Exception e)
{
e.printStackTrace();
return "Fail - " + e.getMessage();
}
                          }
```

186. How to do Applet testing using selenium?

```
// selenium setup
selenium = new DefaultJavaSelenium("localhost",4444, browserString , url);
selenium.start();
selenium.open(url);
// get the appletfixure to control fest JAppletFixture
AppletFixture dialog = selenium.applet(LIST_APPLET_ID)
// fest similar API for autmation testing
dialog.comboBox("domain").select("Users");
dialog.textBox("username").enterText("alex.ruiz");
dialog.button("ok").click();
```

187. If Default port no is busy how to change port no?

```
We can use any port number which is valid.. First create an object to remote control
configuration.
Use 'setPort' method and provide valid port number(4545,5555,5655, etc).. There
after attach this
remote control configuration object to selenium server..i.e
RemoteControlConfiguration r= new RemoteControlConfiguration();
r.setPort(4567);
SeleniumServer s= new SeleniumServer(r);
```

188. I want to find the location of ""b"" in the below code, how can I find out without using xpath, name,id, csslocator, index.

```
<div>
<Button>a</button>
<Button>b</button>
<Button>c</button>
</div>
Ans
driver.findElement(By.xpath("//*[contains(text(),'b')]")).click();
or
//div/button[contains(text(),'b']
```

189.List out the test types that are supported by Selenium?

For web based application testing selenium can be used The test types can be supported are

Functional

Regression

For post release validation with continuous integration automation tool could be used

Jenkins Hudson Quick Build Cruise Control

190.While using click command can you use screen coordinate?

To click on specific part of element, you would need to use clickAt command. clickAt command accepts element locator and x, y co-ordinates as arguments□

clickAt (locator, cordString)

191.Why testers should opt for Selenium and not QTP?

Selenium is more popular than QTP as

Selenium is an open source whereas QTP is a commercial tool

Selenium is used specially for testing web based applications while QTP can be used for testing client server application also

Selenium supports Firefox, IE, Opera, Safari on operating systems like Windows, Mac, Linux etc. however QTP is limited to Internet Explorer on Windows.

Selenium supports many programming languages like Ruby, Perl, Python whereas QTP supports only VB script.

192.What is heightened privileges browsers?

The purpose of heightened privileges is similar to Proxy Injection, allows websites to do something that are not commonly permitted. The key difference is that the browsers are launced in a special mode called heightened privileges. By using these browser mode, Selenium core can open the AUT directly and also read/write its content without passing the whole AUT through the Selenium RC server.

193.What are the features of TestNG and list some of the functionality in TestNG which makes it more effective?

TestNG is a testing framework based on JUnit and NUnit to simplify a broad range of testing needs, from Unit Testing to Integration Testing. And the functionality which makes it efficient testing framework are

Support for annotations

Support for data-driven testing

Flexible test configuration

Ability to re-execute failed test cases

194.Explain what are the JUnits annotation linked with Selenium?

The JUnits annotation linked with Selenium are @Before public void method() – It will perform the method () before each test, this method can prepare the test

@Test public void method() – Annotations @Test identifies that this method is a test method environment

@After public void method()- To execute a method before this annotation is used, test method must start with test @Before.

195.Explain how you can login into any site if it's showing any authentication popup for password and username?

Pass the username and password with url

Syntax-http://username:password@url (http://username:password@url)

ex- http://creyate:tom@www.gmail.com (http://creyate:tom@www.gmail.com)

196.Explain what is the difference between Borland Silk and Selenium?

Silk Test Tool:

Borland Silk test is not a free testing tool

Silk test supports only Internet Explorer and Firefox

Silk test uses test scripting language

Silk test can be used for client server applications

Selenium Test Tool:

Selenium is completely free test automation tool

Selenium supports many browsers like Internet Explorer, Firefox, Safari, Opera and so on

Selenium suite has the flexibility to use many languages like Java, Ruby, Perl and so on

Selenium can be used for only web application

197.Explain how Selenium Grid works?

Selenium Grid sent the tests to the hub. These tests are redirected to Selenium Webdriver, which launch the browser and run the test. With entire test suite, it allows for running tests in parallel.

198.Can we use Selenium grid for performance testing?

Yes. But not as effectively as a dedicated Performance Testing tool like Loadrunner

199. List the advantages of Webdriver over Selenium Server?

If you are using Selenium-WebDriver, you don’t need the Selenium Server as it is using totally different technology

Selenium Server provides Selenium RC functionality which is used for Selenium 1.0 backwards compatibility

Selenium Web driver makes direct calls to browser using each browsers native support for automation, while Selenium RC requires selenium server to inject Javascript into the browser.

200.Mention what are the capabilities of Selenium WebDriver or Selenium 2.0?

WebDriver should be used when requiring improvement support for

Handling multiple frames, pop ups , multiple browser windows and alerts

Page navigation and drag & drop

Ajax based UI elements

Multi browser testing including improved functionality for browser not well supported by Selenium 1.0.

201.While injecting capabilities in webdriver to perform tests on a browser which is not supported by a webdriver what is the limitation that one can come across?

Major limitation of injecting capabilities is that “findElement” command may not work as expected.

202.Explain how you can find broken images in a page using Selenium Web driver?

To find the broken images in a page using Selenium web driver is

Get XPath and get all the links in the page using tag name

In the page click on each and every link

Look for 404/500 in the target page title

203.Which web driver implementation is fastest?

HTMLUnit Driver implementation is fastest, HTMLUnitDriver does not execute tests on browser but plain http request, which is far quick than launching a browser and executing tests

204. What is the command that is used in order to display the values of a variable into the output console or log?

In order to display a constant string, command can be used is echo

If order to display the value of a variable you can use command like echo ${variable name>>

Above is using PHP. If you are using Java, replace echo with System.out.println

205.Explain how you can use recovery scenario with Selenium

Recovery scenarios depends upon the programming language you use. If you are using Java then you can use exception handling to overcome same. By using “Try Catch Block” within your Selenium WebDriver Java tests.

206. Explain how to iterate through options in test script?

To iterate through options in test script you can loop features of the programming language, for example to type different test data in a text box you can use “for” loop in Java

// test data collection in an array

String[ ] testData = { “test1” , “test2” , “test3” } ;

// iterate through each test data

For (string s: test data) { selenium.type (“elementLocator”, testData) ;

207.How can you prepare customized html report using TestNG in hybrid framework?

There are three ways: Junit: With the help of ANT

TestNG: Using inbuilt default.html to get the HTML report. Also XST reports from ANT, Selenium, TestNG combinations.

Using our own customized reports using XSL jar for converting XML content to HTML

208.Explain Cucumber shortly.

Cucumber is a tool that is based on Behavior Driven Development (BDD) methodology. The main aim of the Behavior Driven Development framework is to make various project roles such as Business Analysts, Quality Assurance, Developers, etc., understand the application without diving deep into the technical aspects.

209.What language is used by Cucumber?

Gherkin is the language that is used by the Cucumber tool. It is a simple English representation of the application behavior. Gherkin language uses several keywords to describe the behavior of applications such as Feature, Scenario, Scenario Outline, Given, When, Then, etc.

210.What is meant by a feature file?

A feature file must provide a high-level description of an Application Under Test (AUT). The first line of the feature file must start with the keyword ‘Feature’ followed by the description of the application under test.

A feature file may include multiple scenarios within the same file. A feature file has the extension .feature.

211.What are the various keywords that are used in Cucumber for writing a scenario?

Mentioned below are the keywords that are used for writing a scenario:

Given

When

Then

And

212.What is the purpose of a Scenario Outline in Cucumber?

Scenario outline is a way of parameterization of scenarios. This is ideally used when the same scenario needs to be executed for multiple sets of data, however, the test steps remain the same. Scenario Outline must be followed by the keyword ‘Examples’, which specify the set of values for each parameter.

213.What programming language is used by Cucumber?

Cucumber tool provides support for multiple programming languages such as Java, .Net, Ruby etc. It can also be integrated with multiple tools such as Selenium, Capybara, etc

214.What is the purpose of the Step Definition file in Cucumber?

A step definition file in Cucumber is used to segregate the feature files from the underlying code. Each step of the feature file can be mapped to a corresponding method on the Step Definition file.

While feature files are written in an easily understandable language like, Gherkin, Step Definition files are written in programming languages such as Java, .Net, Ruby, etc.

215.What are the major advantages of the Cucumber framework?

Given below are the advantages of the Cucumber Gherkin framework that make Cucumber an ideal choice for rapidly evolving Agile methodology in today's corporate world.

1. Cucumber is an open-source tool.
2. Plain Text representation makes it easier for non-technical users to understand the scenarios.
3. It bridges the communication gap between various project stakeholders such as Business Analysts, Developers, and Quality Assurance personnel.
4. Automation test cases developed using the Cucumber tool are easier to maintain and understand as well.
5. Easy to integrate with other tools such as Selenium and Capybara.

216.Provide an example of a feature file using the Cucumber framework

Following is an example of a feature file for the scenario 'Login into the application:

Feature: Login to the application under test. Scenario:

1. Login to the application.
2. Open the Chrome browser and launch the application.
3. When the user enters the username onto the UserName field.
4. And User enters the password into the Password field.
5. When the user clicks on the Login button.
6. Then validate if the user login is successful.

217.Provide an example of a Scenario Outline using the Cucumber framework.

The following is an example of a Scenario Outline keyword for the scenario 'Upload a file'. The number of parameter values to be included in the feature file is based on the tester's choice.

Scenario Outline: 1. Upload a file

2. Given that the user is on upload file screen.
3. When a user clicks on the Browse button.
4. And user enters onto the upload textbox.
5. And user clicks on the enter button.
6. Then verify that the file upload is successful.

Example:

|filename| |file1| |file2

218.What is the purpose of the Behaviour Driven Development (BDD) methodology in the real world?

BDD is a methodology to understand the functionality of an applicatio the simple plain text representation.

The main aim of the Behavior Driven Development framework is to make various project roles such as Business Analysts, Quality Assurance, Developers, and Support Teams understand the application without diving deep into the technical aspects.

219. What is the limit for the maximum number of scenarios that can be included in the feature file?

A feature file can contain a maximum of 10 scenarios, but the number can vary from project to project and from one organization to another. But it is generally advisable to limit the number of scenarios included in the feature file.

220. What is the use the of Background keyword in Cucumber?

Background keyword is used to group multiple given statements into a single group. This is generally used when the same set of the given statement is repeated in each scenario of the feature file.

221. What symbol is used for parameterization in Cucumber?

Pipe symbol (|) is used to specify one or more parameter values in a feature file.

222.What is the purpose of Examples keyword in Cucumber?

Examples keyword is used to specify values for each parameter used in the scenario. Scenario Outline keyword must always be followed by the keyword Examples.

223.Provide an example of a step definition file in Cucumber.

```
Step definition corresponding to the step “Open Chrome browser and launch the
application” may look like the code mentioned below:
@Given("^Open Chrome browser and launch the application$")
public void openBrowser()
{
driver = new ChromeDriver();
driver.manage().window().maximize();
driver.get("www.facebook.com");
}
```

224.What is the purpose of the Cucumber Options tag?

Cucumber Options tag is used to provide a link between the feature files and step definition files. Each step of the feature file is mapped to a corresponding method on the step definition file

Below is the syntax of Cucumber Options tag:

@CucumberOptions(features="Features",glue={"StepDefinition"})

225.How can Cucumber be integrated with Selenium WebDriver?

Cucumber can be integrated with the Selenium Webdriver by downloading the necessary JAR files.

Given below are the list of JAR files that are to be downloaded for using Cucumber with Selenium web driver:

cucumber-core-1.2.2.jar

cucumber-java-1.2.2.jar

cucumber-junit-1.2.2.jar

cucumber-jvm-deps-1.0.3.jar

cucumber-reporting-0.1.0.jar

gherkin-2.12.2.jar

226. When is Cucumber used in real-time?

Cucumber tool is generally used in real-time to write acceptance tests for an application. It is generally used by non-technical people such as Business Analysts, Functional Testers, etc.

227.What is the use of Behavior Driven Development in Agile methodology?

The advantages of Behavior Driven Development are best realized when non-technical users such as Business Analysts use BDD to draft requirements and provide the same to the developers for implementation.

In Agile methodology, user stories can be written in the format of feature file and the same can be taken up for implementation by the developers

228.Explain the purpose of keywords that are used for writing a scenario in Cucumber.

“Given” keyword is used to specify a precondition for the scenario.

“When” keyword is used to specify an operation to be performed.

“Then” keyword is used to specify the expected result of a performed action.

“And” keyword is used to join one or more statements together into a single statement.

229.What is the name of the plugin that is used to integrate Eclipse with Cucumber?

Cucumber Natural Plugin is the plugin that is used to integrate Eclipse with Cucumber.

230.What is the meaning of the TestRunner class in Cucumber?

TestRunner class is used to provide the link between the feature file and the step definition file. The next question provides a sample representation of how the TestRunner class will look like. A TestRunner class is generally an empty class with no class definition.

231. Provide an example of the TestRunner class in Cucumber.

```
Package com.sample.TestRunner
importorg.junit.runner.RunWith;
importcucumber.api.CucumberOptions;
importcucumber.api.junit.Cucumber;
@RunWith(Cucumber.class)
@CucumberOptions(features="Features",glue={"StepDefinition"})
public class Runner
{
}
```

232.What is the starting point of execution for feature files?

When integrated with Selenium, the starting point of execution must be from the TestRunner class.

233. Should any code be written within the TestRunner class?

No code should be written under the TestRunner class. It should include the tags @RunWith and @CucumberOptions.

234. What is the use of features property under the Cucumber Options tag?

Features property is used to let the Cucumber framework identify the location of the feature files.

235.What is the use of glue property under the Cucumber Options tag?

Glue property is used to let the Cucumber framework identify the location of step definition files.

236. What is Cucumber? Why is it used?

Cucumber is a testing tool based on Behavior Driven Development (BDD) framework. It is used to run functional tests written in plain text and develop test cases for software functionality. It plays a supporting role in automated testing.

In other words, we can say that "Cucumber is a software tool used by the testers to develop test cases for the testing of behavior of the software."

237.What are the two files required to execute a Cucumber test scenario?

Following are the two files required to execute a Cucumber test scenario:

```
Features
```

Step Definition

238.What are the differences between Jbehave and Cucumber?

Although Cucumber and Jbehave are designed for the same purpose, the most distinctive difference between them is that Jbehave is based on stories while Cucumber is based on features

239.What do you understand by test harness in Cucumber?

In Cucumber, the test harness allows for separating responsibility between setting up the context and interacting with the browser, and cleaning up the step definition files. It collects stubs, drivers, and other supporting tools required to automate test execution in testing.

240.What is the difference between RSpec and Cucumber? When should we use RSpec and when to use Cucumber?

RSpec and Cucumber both are the example of testing frameworks. RSpec uses traditional Unit Testing. It means it uses testing a class or part of the application in isolation from the rest of the application. So your model does what your model is supposed to do, the controller does what it is supposed to do, etc. RSpec and Cucumber both are used for Acceptance Testing, also called ATDD, BDD, etc.

Difference between RSpec and Cucumber

The main difference between RSpec and Cucumber is the business readability factor. RSpec is mainly used for Unit Testing. On the other hand, Cucumber is mainly used for Behavior-driven development. We can also use it for System and Integration Testing. In Cucumber, the specifications or features are separate from the test code, so the product owners can provide or review the specification without going through the code. These are the .feature files that you make in Cucumber.

RSpec also has a similar mechanism, but instead of describing a step with a Describe or Context, it uses the business specification to execute that statement. This approach is a little easier for developers to work with but a bit harder for non-technical guys.

Which should we use?

For a core developer, it is the best choice to use RSpec. It is easier to understand for a technical person and offers a few advantages in keeping things scoped and under control because you don't have to mess up with RegExs for test steps.

If you are building this for a client, you should choose Cucumber for Acceptance Testing and use RSpec

241.What is the difference between Selenium and Cucumber?

functional testing. But there are some differences between them.

Following are some critical differences between Selenium and Cucumber:

➢ Selenium is a web browser automation tool for web apps, while Cucumber is an automation tool for behavior-driven development that can be used with Selenium (or Appium).

➢ Selenium is used for automated UI testing, while Cucumber is used for acceptance testing.

➢ Selenium is preferred by technical teams (SDETs/programmers), while Cucumber is typically preferred by non-technical teams (business stakeholders and testers).

➢ Selenium can work independently of Cucumber. Cucumber depends on Selenium or Appium for step-definition implementation.

➢ In Selenium, the script creation is complex, while Cucumber is simpler than Selenium.

242.Why we have to use Cucumber with Selenium?

Cucumber and Selenium are both testing frameworks and prevalent technologies. Many organizations use Selenium for functional testing. Along with Selenium, these organizations integrate Cucumber with Selenium as Cucumber makes it easy to read and understand the application flow. The most significant benefit of using Cucumber with Selenium is that it facilitates developers to write test cases in simple feature files easily understood by managers, non-technical stakeholders, and business analysts.

provides the facility to write tests in a human-readable language called Gherkin. The Selenium-Cucumber framework supports programming languages such as Java, .NET, PHP, Python, Perl, etc.

243.) What do you understand by TDD, and what are the different processes used in TDD?

TDD is an acronym that stands for Test-Driven Development. This is a software development technique used to create the test cases first and then write the code underlying those test cases. Although TDD is a development technique, it can also be used for automation testing development. TDD takes more time for development because it tends to find very few defects. The result provided by the TDD development technique has improved the quality of code, and that can be more reusable and flexible. TDD also helps developers to achieve high test coverage of about 90-100%. The only disadvantage for developers following TDD is to write their test cases before writing the code . Following is the list of simple 6 step process used by TDD methodology:

First, write the test case: You have to write an automated test case according to your requirements.

Run all the test cases: Now, run these automated test cases on the currently developed code.

Develop the code for that test case: In this process, you must write the code to make that test case work as expected if the test case fails.

Run test cases again: Now, you have to rerun the test cases and check if all the test cases developed so far are implemented.

Refactor your code: This is an optional step. But, it is advised to refactor your code to make it more readable and reusable. That's why it is essential.

Repeat steps 1- 5 for new test cases: This is the last step. Here, you have to repeat the cycle for the other

244.What are the similarities between BDD and TDD?

➢ TDD stands for Test-Driven Development, and BDD stands for Behavior Driven Development. Both are two software development techniques.

➢ BDD and TDD are both very similar as they are both testing strategies for a software application. In both cases, the developers have to write the test before writing the code to pass the test. The second main similarity between them is in both cases; the tests can be used as part of an automated testing framework to prevent bugs.

245.What is Functional Testing ?

1. Testing the features and operational behavior of a product to ensure they corre□spond to its specifications.
2. Testing that ignores the internal mechanism of a system or component and focus□es solely on the outputs generated in response to selected inputs and execution conditions.

Goal of Functional Testing :

The goal of Functional Testing is to verify whether your product meets the intended functional specifications mentioned in your development documentation.

Testing Activities Included :

Test all the links in your webpages are working correctly and make sure there are no broken links.

Links to be checked will include -

1. Outgoing links
2. Internal links
3. Anchor Links
4. MailTo Links

Types of Web App Testing

Test Forms are working as expected. This will include□01. Scripting checks on the form are working as expected. For example- if a user does not fill a mandatory field in a form an error message is shown.

2. Check default values are being populated.
3. Once submitted, the data in the forms is submitted to a live database or is linked to a working email address.
4. Forms are optimally formatted for better readability.

Test business workflow- This will include

1. Testing your end - to - end workflow/ business scenarios which takes the user through a series of web pages to complete.
2. Test negative scenarios as well, such that when a user executes an unexpected step, appropriate error message or help is shown in your web application.

Functional Test Scenarios :

1. Test all the mandatory fields should be validated.

2. Test the asterisk sign should display for all the mandatory fields.
3. Test the system should not display the error message for optional fields.
4. Test that leap years are validated correctly & do not cause errors/miscalcula□Test HTML and CSS to ensure that search engines can crawl your site easily. This will include
5. Checking for Syntax Errors
6. Readable Color Schemas
7. Standard Compliance. Ensure standards such W3C, OASIS, IETF, ISO, ECMA, or WS-I are followed.

Test business workflow- This will include

1. Testing your end - to - end workflow/ business scenarios which takes the user through a series of web pages to complete.
2. Test negative scenarios as well, such that when a user executes an unexpected step, appropriate error message or help is shown in your web application.

Functional Test Scenarios : 01. Test all the mandatory fields should be validated.

2. Test the asterisk sign should display for all the mandatory fields.
3. Test the system should not display the error message for optional fields.
4. Test that leap years are validated correctly & do not cause errors/miscalculations.
5. Test the numeric fields should not accept the alphabets and proper error message should display.
6. Test for negative numbers if allowed for numeric fields.
7. Test division by zero should be handled properly for calculations.
8. Test the max length of every field to ensure the data is not truncated.
9. Test the pop up message (“This field is limited to 500 characters”) should dis□play if the data reaches the maximum size of the field.
10. Test that a confirmation message should display for update and delete opera□tions.
11. Test the amount values should display in currency format.
12. Test all input fields for special characters.
13. Test the timeout functionality.
14. Test the Sorting functionality. Test the Sorting functionality.
15. Test the functionality of the buttons available.
16. Test the Privacy Policy & FAQ is clearly defined and should be available for users.
17. Test if any functionality fails the user gets redirected to the custom error page.

246.What is Usability Testing ?

1. Usability testing is nothing but the User-friendliness check.
2. In Usability testing, the application flow is tested so that a new user can un□derstand the application easily.
3. Basically, system navigation is checked in Usability testing.

Goal of Usability Testing :

A Usability test establishes the ease of use and effectiveness of a product using a standard Usability test practices.

Testing Activities Included :

Test the site Navigation:

1. Menus, buttons or Links to different pages on your site should be easily visible and consistent on all webpages.
2. Test the Content.
3. Content should be legible with no spelling or grammatical errors.
4. Images if present should contain an “alt” text.

Usability Test Scenarios :

1. Web page content should be correct without any spelling or grammatical errors.

2. All fonts should be same as per the requirements.
3. All the text should be properly aligned.
4. All the error messages should be correct without any spelling or grammatical errors and the error message should match with the field label.
5. Tool tip text should be there for every field.
6. All the fields should be properly aligned.
7. Enough space should be provided between field labels, columns, rows, and error messages.
8. All the buttons should be in a standard format and size.
9. Home link should be there on every single page.
10. Disabled fields should be grayed out.
11. Check for broken links and images.
12. Confirmation message should be displayed for any kind of update and delete op□eration.
13. Check the site on different resolutions (640 x 480, 600x800 etc.?)
14. Check the end user can run the system without frustration.
15. If there is an error message on submit, the information filled by the user should be there.
16. Title should display on each web page
17. All fields (Textbox, dropdown, radio button etc) and buttons should be accessible by keyboard shortcuts and the user should be able to perform all operations by using keyboard.
18. Check if the dropdown data is not truncated due to the field size and also check whether the data is hardcoded or managed via administrator.

247.Interface Testing

Three areas to be tested here are - Application, Web and Database Server

1. Application: Test requests are sent correctly to the Database and output at the client side is displayed correctly. Errors if any must be caught by the application and must be only shown to the administrator and not the end user.
2. Web Server: Test Web server is handling all application requests without any service denial.
3. Database Server: Make sure queries sent to the database give expected results. Test system response when connection between the three layers (Application, Web and Database) cannot be established and appropriate message is shown to the end user.

248.Database Testing

Database is one critical component of your web application and stress must be laid to test it thoroughly. Testing activities will include□01. Test if any errors are shown while executing queries.

2. Data Integrity is maintained while creating, updating or deleting data in data□base .
3. Check response time of queries and fine tune them if necessary.
4. Test data retrieved from your database is shown accurately in your web appli□cation.

To perform the Database testing, the tester should be aware of the below men□tioned points:

1. The tester should understand the functional requirements, business logic, ap□plication flow and database design thoroughly.
2. The tester should figure out the tables, triggers, store procedures, views and cursors used for the application.
3. The tester should understand the logic of the triggers, store procedures, views and cursors created.
4. The tester should figure out the tables which get affected when insert update and delete (DML) operations are performed through the web or desktop appli□cations.

Test Scenarios for Database Testing

1. Verify the database name: The database name should match with the specifica□tions.
2. Verify the Tables, columns, column types and defaults. All things should match with the specifications.
3. Verify whether the column allows a null or not.

4. Verify the Primary and foreign key of each table.
5. Verify the Stored Procedure.
6. Test whether the Stored procedure is installed or not.
7. Verify the Stored procedure name
8. Verify the parameter names, types and number of parameters.
9. Test the parameters if they are required or not.
10. Test the stored procedure by deleting some parameters
11. Test when the output is zero, the zero records should be affected.
12. Test the stored procedure by writing simple SQL queries.
13. Test whether the stored procedure returns the values
14. Test the stored procedure with sample input data.
15. Verify the behavior of each flag in the table.
16. Verify the data gets properly saved into the database after the each page submi□sion.
17. Verify the data if the DML (Update, delete and insert) operations are performed.
18. Check the length of every field. The field length in the back end and front end must be same.
19. Verify the database names of QA, UAT and production. The names should beunique.
20. Verify the encrypted data in the database.
21. Verify the database size. Also test the response time of each query executed.
22. Verify the data displayed on the front end and make sure it is same in the back end.
23. Verify the data validity by inserting the invalid data in the database.
24. Verify the Triggers

249.What is Compatibility Testing

In [ ]:
```
Compatibility testing is used to determine if your software is compatible with other
elements of a system with which it should operate, e.g. Browsers, Operating Sys•tems, c


Goal of Compatibility Testing :

The purpose of Compatibility testing is to evaluate how well software performs in
a particular browser, Operating Systems, hardware or software. Compatibility tests
ensure that your web application displays correctly across different devices.


Testing Activities Included :


Browser Compatibility Test: Same website in different browsers will display dif•ferentl
browsers, JavaScript, AJAX and authentication is working fine. You may also check
for Mobile Browser Compatibility.

The rendering of web elements like buttons, text fields etc. changes with change
in Operating System. Make sure your website works fine for various combination
of Operating systems such as Windows, Linux, Mac and Browsers such as Firefox,
Internet Explorer, Safari etc.


Compatibility Test Scenarios :


01. Test the website in different browsers (IE, Firefox, Chrome, Safari and Opera)
and ensure the website is displaying properly.

02. Test the HTML version being used is compatible with appropriate browser
versions.

03. Test the images display correctly in different browsers.

04. Test the fonts are usable in different browsers.

05. Test the java script code is usable in different browsers.

06. Test the Animated GIF's across different browsers
```

250.What is Performance Testing ?

In [ ]:
```
Performance Testing is conducted to evaluate the compliance of a system or
component with specified performance requirements.

Testing Activities Included :

01. Website application response times at different connection speeds.

02. Load test your web application to determine its behavior under normal and
peak loads.

03. Stress test your web site to determine its break point when pushed to be•yond norma

04. Test if a crash occurs due to peak load, how does the site recover from such
an event.

05. Make sure optimization techniques like gzip compression, browser and
server side cache enabled to reduce load times.



General Test Scenarios :

01. To determine the performance, stability and scalability of an application
under different load conditions.

02. To determine if the current architecture can support the application at
peak user levels.

03. To determine which configuration sizing provides the best performance
level.

04. To identify application and infrastructure bottlenecks.

05. To determine if
the new version of the software adversely had an impact
on response time.

06. To evaluate product and/or hardware to determine if it can handle project•ed load v
```

251.Security Testing

Security Testing involves the test to identify any flaws and gaps from a security point of view.

Test Scenarios for Security Testing :

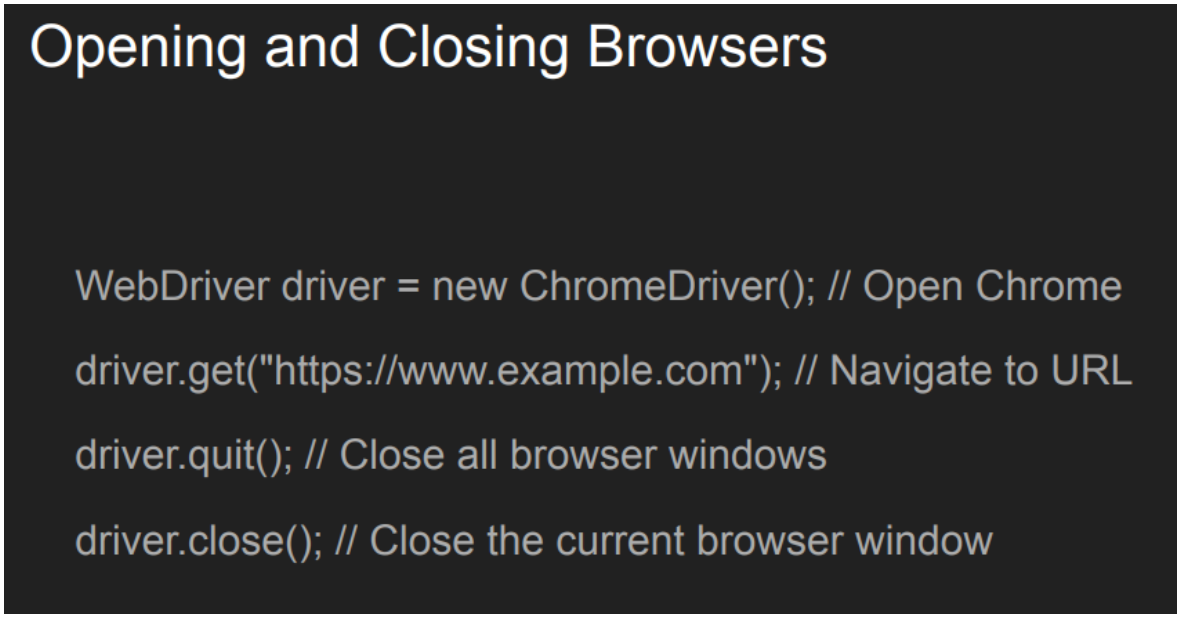
1. Verify the web page which contains important data like password, credit card numbers, secret answers for security question etc should be submitted via HTTPS (SSL).
2. Verify the important information like password, credit card numbers etc should display in encrypted format.
3. Verify password rules are implemented on all authentication pages like Regis□tration, forgot password, change password.
4. Verify if the password is changed the user should not be able to login with the old password.
5. Verify the error messages should not display any important information.
6. Verify if the user is logged out from the system or user session was expired, the user should not be able to navigate the site.
7. Verify to access the secured and non secured web pages directly without login.
8. Verify the “View Source code” option is disabled and should not be visible to the user.
9. Verify the user account gets locked out if the user is entering the wrong pass□word several times.
10. Verify the cookies should not store passwords.
11. Verify if, any functionality is not working, the system should not display any ap□plication, server, or database information. Instead, it should display the custom error page.

12. Verify the SQL injection attacks.
13. Verify the user roles and their rights. For Example The requestor should not be able to access the admin page.
14. Verify the important operations are written in log files, and that information should be traceable.
15. Verify the session values are in an encrypted format in the address bar.
16. Verify the cookie information is stored in encrypted format.
17. Verify the application for Brute Force Attacks

DIFFERENCES BETWEEN TEST **PLAN** TEST **SUİTE** TEST **CASE** TEST **SCENARİOS**

TEST PLAN	*TEST SUİTE*	*TEST CASE*	*TEST SCENARİOS*
1. Test plan is a document that defines the scope, objective, and strategy of testing.	1. Prepared after test plan, test suite consists of a collection of test cases.	1. Test case is an important document that consists of various crucial details about testing.	1. Test scenarios or test condition is any functionality of the software that can be tested.
2. It is of three types, level specific, type specific, master test plan.	2. It is of two types, abstract and executable test suites.	2. These are of two types, **formal test cases and informal test case**.	2. It is performed from the perspective of the end users.
3. It follows a standardized template, which offers details about the testing process.	3. Test suites define the objective and goal of test cases that are intended to test the software product.	3. It defines a set of **conditions** that help verify the compliance of the software with specified functionality.	3. It defines the various operations that are performed by the team on the software product.
4. It is derived from Product Description, **Software Requirement Specifications (SRS)**, or Use Case Document.	4. Separate test suites offer great advantages to team and it created testing to be hassle free, flexible, **agile**.	4. These are derived from the test scenarios and are designed on the basis of Software Requirement Specification (SRS).	4. They are derived from the **use cases** and they ensure complete test coverage.

Selenium keywords



Element Interaction

```
driver.findElement(By.id("username")).sendKeys("testuser"); // Enter text
driver.findElement(By.name("submit")).click(); // Click a button
driver.findElement(By.linkText("Home")).click(); // Click a link
driver.findElement(By.xpath("//div[@class='message']")).getText(); // Get text
```

Navigation

```
driver.navigate().to("https://www.newsite.com"); // Navigate to a new URL
driver.navigate().back(); // Go back to the previous page
driver.navigate().forward(); // Go forward to the next page
driver.navigate().refresh(); // Refresh the current page
```

Window and Frame Handling

```
driver.switchTo().window("windowName"); // Switch to a specific window
driver.switchTo().frame("frameId"); // Switch to a frame
driver.switchTo().defaultContent(); // Switch back to the main content
```

Browser Management

```
driver.manage().window().maximize(); // Maximize the browser window
driver.manage().window().fullscreen(); // Set full-screen mode
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS); // Set implicit wait
```

Mouse and Keyboard Actions

```
Actions actions = new Actions(driver);
actions.moveToElement(element).click().build().perform(); // Move to and click
actions.doubleClick(element).perform(); // Double-click
actions.dragAndDrop(source, target).perform(); // Drag and drop
actions.sendKeys(element, "text").perform(); // Send keystrokes
```

Assertions

```
Assert.assertTrue(element.isDisplayed()); // Assert element visibility
Assert.assertEquals(driver.getTitle(), "Expected Title"); // Assert title
```

By default these are hard assertions, means if these fail then tests will be exited or stopped

Soft Assertions

```
SoftAssert softassert = new SoftAssert();
 //Soft assert applied to verify title
 softassert.assertEquals(ActualTitle, ExpectedTitle);
 softassert.assertAll();
```

By default these assertions, do not stop if these assertions fail & are reported in your test reports

Explicit Wait

```
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.elementToBeClickable(element)); // Explicit wait
```

These are conditional waits & can be applied to satisfy a particular condition, then continue test execution if condition met or failed if not met in mentioned amount of time

Implicit Wait

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

Once the command is run, Implicit Wait remains for the entire duration for which the browser is open. It's default setting is 0, and the specific wait time needs to be set by the following protocol.

Implicit wait increases test script execution time

Fluent Wait - looks for a web element repeatedly at regular intervals until timeout happens or until the object is found

```
FluentWait wait = new FluentWait(driver);
//Specify the timeout of the wait
wait.withTimeout(5000, TimeUnit.MILLISECONDS);
//Specify polling time
wait.pollingEvery(250, TimeUnit.MILLISECONDS);
//Specify what exceptions to ignore
wait.ignoring(NoSuchElementException.class)
//This is how we specify the condition to wait on.
wait.until(ExpectedConditions.alertIsPresent());
```

Alerts

```
Alert alert = driver.switchTo().alert();
alert.accept(); // Accept an alert
alert.dismiss(); // Dismiss an alert
alert.sendKeys("text"); // Send text to an alert
```

Getting Page Information

```
driver.getTitle(); // Get the page title
driver.getCurrentUrl(); // Get the current URL
driver.getPageSource(); // Get the entire page source code
```

Element Information

```
element.isDisplayed(); // Check if an element is visible
element.isEnabled(); // Check if an element is enabled
element.isSelected(); // Check if an element is selected
element.getAttribute("attribute_name"); // Get an element's attribute value
element.getTagName(); // Get an element's tag name
```

Menu Handling

```
Select select = new Select(element); // Create a Select object for dropdowns
select.selectByIndex(1); // Select an option by index
select.selectByValue("option2"); // Select an option by value
select.selectByVisibleText("Option 3"); // Select an option by visible text
```

Keyboard Actions

```
actions.keyDown(Keys.CONTROL).sendKeys("a").keyUp(Keys.CONTROL).perform(); // Ctrl+A

actions.keyDown(Keys.SHIFT).sendKeys("hello").keyUp(Keys.SHIFT).perform(); // Type "HELLO"
actions.sendKeys(Keys.ENTER).perform(); // Press Enter
```

Mouse Actions

```
actions.contextClick(element).perform(); // Right-click
actions.clickAndHold(element).perform(); // Click and hold
actions.release(element).perform(); // Release the mouse button
```

Taking Screenshots

```
File screenshot = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
```

Switching to Iframes

driver.switchTo().frame("iframe_name");

Scrolling

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("window.scrollTo(0, 1000)");

Actions actions = new Actions(driver);

// Scroll down a specific distance:

actions.moveToElement(element).clickAndHold().moveByOffset(0, 500).release().build().perform();

Cookie Management

Get all cookies: Set<Cookie> cookies = driver.manage().getCookies();

Get a specific cookie: Cookie cookie = driver.manage().getCookieNamed("cookie_name");

Add a cookie: driver.manage().addCookie(new Cookie("cookie_name", "cookie_value"));

Delete a cookie: driver.manage().deleteCookieNamed("cookie_name");

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

Alert Handling (Extended)

Dismiss an alert: alert.dismiss();

Get alert text: String alertText = alert.getText();

Page Source Interactions

Find elements in page source: List<WebElement> elements = driver.findElements(By.xpath("//*"));

Execute JavaScript code: JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("document.getElementById('myElement').click();");

Advanced Navigation

Refresh the page with hard reload: driver.navigate().refresh(true);

Navigate to a page relative to the current URL: driver.navigate().to("new_page.html");

Logging and Reporting

Enable logging: System.setProperty("webdriver.chrome.logfile", "chromedriver.log");

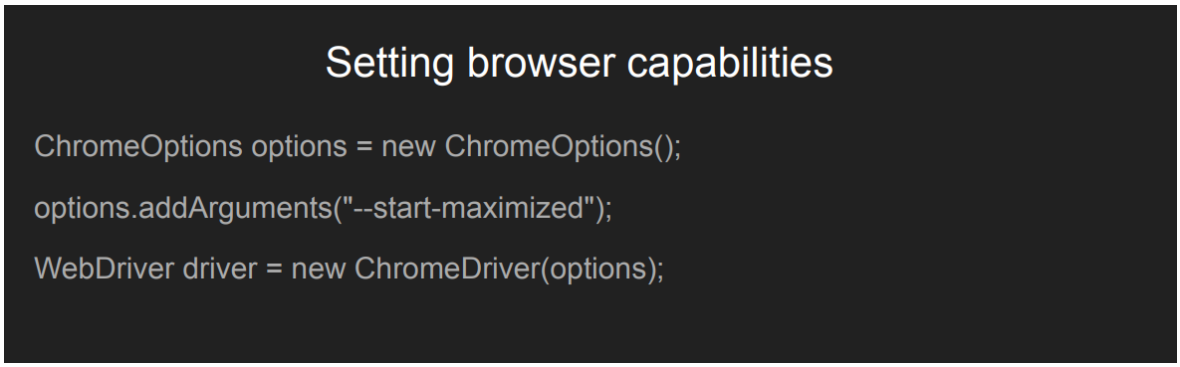
Create custom test reports: Utilize third-party libraries like ExtentReports or Allure or TestNG reports

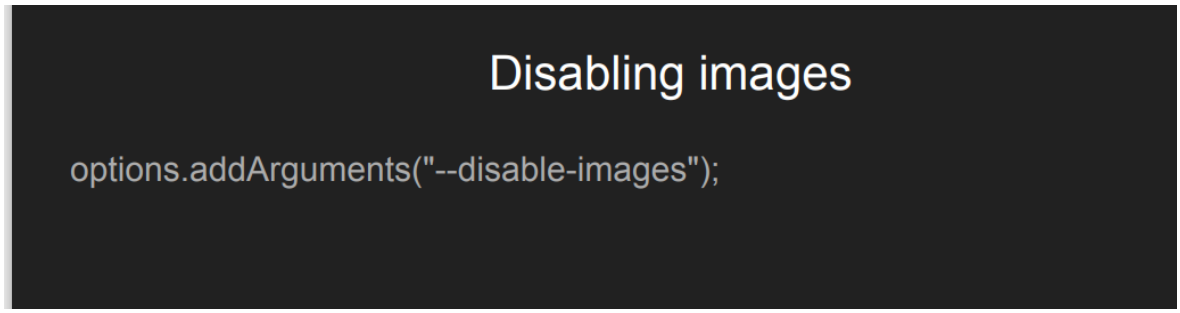
Headless Mode

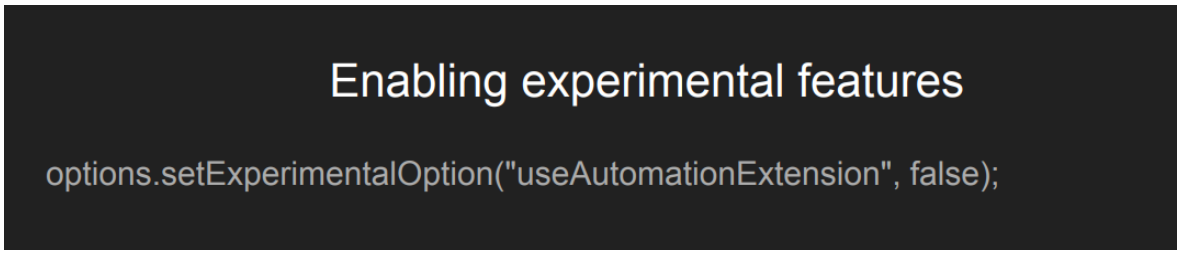
Run tests without a visible browser: ChromeOptions options = new ChromeOptions();

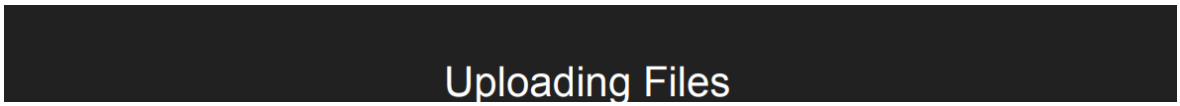
options.addArguments("--headless");

WebDriver driver = new ChromeDriver(options);


Setting browser capabilities
ChromeOptions options = new ChromeOptions();
options.addArguments("--start-maximized");
WebDriver driver = new ChromeDriver(options);


Disabling images
options.addArguments("--disable-images");


Enabling experimental features
options.setExperimentalOption("useAutomationExtension", false);


Uploading Files

In [ ]: