**Selenium WebDriver FAQ’s:**

**1**. What is webdriver?

 WebDriver is a simpler, more concise programming interface in addition to addressing some limitations in  the Selenium-RC API. Selenium-WebDriver was developed to better support dynamic web pages where elements of a [page](javascript:void(0);) may change without the page itself being reloaded. WebDriver’s goal is to supply a well-designed object-oriented API that provides improved support for modern advanced web-app testing problems.

**2**.      What are the advantages of selenium2.0/webdriver?

* Need no server to start
* Easy to code
* Has sophisticated API to support wide verity of browsers.
* Supports to test dynamic UI web apps.

**3**.      Difference between the selenium1.0 and selenium 2.0?

|  |  |
| --- | --- |
| **Selenium 1.0** | **Selenium 2.0/Webdriver** |
| 1.      It ‘injected’ javascript functions into the browser when the browser was loaded and then used its javascript to drive the AUT within the browser.  2.      Selenium server need to start to run tests  3.      Has some loop-holes in supporting complex UI apps,Javascript security  4.      No support for headless broswers | 1.      WebDriver makes direct[calls to](javascript:void(0);) the browser using each browser’s native support for automation  2.      Not needed unless tests are run on local machine.  3.      Supports even drag and drop features and no security loop-holes  4.      Supports htmlunit driver –headless browser runs fast |

|  |  |
| --- | --- |
|  |  |

**4**.      What are the Locators are there in selenium 2.0?

It supports locators based on Id,name,xpath,dom,css,class,tagname

**5**.      How to handle the Ajax Applications in Web driver?

There are 2 types of waits webdriver supports to handle ajax applications to make webdrive sync to execution:

**Implicit wait :**

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

**Explicit wait:**   WebDriverWait, FluentWait

WebElement strr = (new WebDriverWait(driver,30)).until(ExpectedConditions.presenceOfElementLocated(By.xpath("//\*[starts-with(@id,'yui\_3\_4\_0\_1\_137509')]/ul/li[2]/a")));

This link explains better about handling ajax in webdriver.

<http://qeworks.com/handle-ajax-call-selenium-webdriver/>

**5**.      How to handle the multiple windows in web driver?

            driver.switchTo().window(Window ID);

**6**.      Difference between findelement() and findelements()?

     findELement will find the first matching element.  
  
     findELements will all the matching elements. You'll probably need to loop through all the elements  returned.

**7**.      How to handle the alerts in web driver?

driver.switchTo().alert().accept();

**8**.      How to take the screen shots in seelnium2.0?

            File src2 = ((TakesScreenshot)driver).getScreenshotAs(OutputType.*FILE*);

            FileUtils.*copyFile*(src2,**new** File("d:\\sc2.jpeg"));

**10**.  What is the limitations of web driver?

* Can not automate [desktop](javascript:void(0);) applications, supports only [web applications](javascript:void(0);)
* No inbuilt commands to generate good reports
* Cannot support readily for new browsers

**11**. What is Selenium 2.0? I have heard this buzz word many times.

Selenium 2.0 is consolidation of two web testing tools – Selenium RC and WebDriver, which claims to give best of both words – Selenium and WebDriver. Selenium 2.0 was officially released only of late.

**12**. Why are two tools being combined as Selenium 2.0, what’s the gain?

Selenium 2.0 promises to give much cleaner API then Selenium RC and at the same time not being restricted by java script Security restriction like same origin policy, which have been haunting Selenium from long. Selenium 2.0 also does not warrant you to use Selenium Server.

**13**. So everyone is going to use Selenium 2.0?

Well no, for example if you are using Selenium Perl client driver than there is no similar offering from Selenium 2.0 and you would have to stick to Selenium 1.0 till there is similar library available for Selenium 2.0

**14**. So how do I specify my browser configurations with Selenium 2.0?

Selenium 2.0 offers following browser/mobile configuration –

·         AndroidDriver,

·         ChromeDriver,

·         EventFiringWebDriver,

·         FirefoxDriver,

·         HtmlUnitDriver,

·         InternetExplorerDriver,

·         IPhoneDriver,

·         IPhoneSimulatorDriver,

·         RemoteWebDriver

And all of them have been implemented from interface WebDriver. To be able to use any of these drivers you need to instantiate their corresponding class.

**15**.How is Selenium 2.0 configuration different than Selenium 1.0?

In case of Selenium 1.0 you need Selenium jar file pertaining to one library for example in case of java you need java client driver and also Selenium server jar file. While with Selenium 2.0 you need language binding (i.e. java, C# etc) and Selenium server jar if you are using Remote Control or Remote WebDriver.

**16**.Can you show me one code example of setting Selenium 2.0?

Here is java example of initializing firefox driver and using Google Search engine –

**protected** WebDriver webDriver;

     //@BeforeClass(alwaysRun=true)

**public** **void** startDriver(){

     webDriver = **new** FirefoxDriver();

     // Get Google search page and perform search on term “Test”

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

     webDriver.findElement(By.*name*("q")).sendKeys("Test");

     webDriver.findElement(By.*name*(“btnG”)).click();

**17**.Which web driver implementation is fastest?

HTMLUnitDriver. Simple reason is HTMLUnitDriver does not execute tests on browser but plain http request – response which is far quick than launching a browser and executing tests. But then you may like to execute tests on a real browser than something running behind the scenes

**18**.What all different element locators are available with Selenium 2.0?

Selenium 2.0 uses same set of locators which are used by Selenium 1.0 – id, name, css, XPath but how Selenium 2.0 accesses them is different. In case of Selenium 1.0 you don’t have to specify a different method for each locator while in case of Selenium 2.0 there is a different method available to use a different element locator. Selenium 2.0 uses following method to access elements with id, name, css and XPath locator –

*driver.findElement(By.id("HTMLid"));*

*driver.findElement(By.name("HTMLname"));*

*driver.findElement(By.cssSelector("cssLocator"));*

*driver.findElement(By.xpath("XPathLocator));*

**19**.How do I submit a form using Selenium?

You can use “submit” method on element to submit form –

element.submit();

Alternatively you can use click method on the element which does form submission.

**20**.Can I simulate pressing key board keys using Selenium 2.0?

You can use “sendKeys” command to simulate key board keys as –

            element.sendKeys(" and some", Keys.ARROW\_UP);

You can also use “sendKeys” to type in text box as –

            HTMLelement.sendKeys("testData");

**21**.How do I clear content of a text box in Selenium 2.0

You can use “clear” method on text box element to clear its content –

            textBoxElement.clear();

**22**.How do I select a drop down value using Selenium2.0?

To select a drop down value, you first need to get the select element using one of element locator and then you can select element using visible text –

Select selectElement = **new** Select(driver.findElement(By.*cssSelector*("cssSelector")));

          selectElement.selectByVisibleText("India");

**23**.What are offering to deal with popup windows while using Selenium 2.0?

You can use “switchTo” window method to switch to a window using window name. There is also one method “getWindowHandles” which could be used to find all Window handles and subsequently bring control on desired window using window handle –

webDriver.switchTo().window("windowName");

**for** (String handle : driver.getWindowHandles()) {

              driver.switchTo().window(handle);

          }

**24**.How about handling frames using Selenium 2.0?

You can use “switchTo” frame method to bring control on an HTML frame –

            driver.switchTo().frame("frameName");

You can also use index number to specify a frame –

            driver.switchTo().frame("parentFrame.4.frameName");

This would bring control on frame named – “frameName” of the 4th sub frame names “parentFrame”

**25**.Can I navigate back and forth in a browser in Selenium 2.0?

You can use Navigate interface to go back and forth in a page. Navigate method of WebDriver interface returns instance of Navigation. Navigate interface has methods to move back, forward as well as to refresh a page –

driver.navigate().forward();

driver.navigate().back();

driver.navigate().refresh();

**26**.What is the order of fastest browser implementation for WebDriver?

HTMLUnitDriver is the fastest browser implementation as it does not involves interaction with a browser, This is followed by Firefox driver and then IE driver which is slower than FF driver and runs only on Windows.

**27**.Is it possible to use Selenium RC API with Selenium 2.0?

You can emulate Selenium 1.0 API with Selenium 2.0 but not all of Selenium 1.0 methods are supported. To achieve this you need to get Selenium instance from WebDriver and use Selenium methods. Method executions might also be slower while simulating Selenium 1.0 with in Selenium 2.0

**28**.Can you show me one example of using Selenium 1.0 in Selenium 2.0?

Code Sample:

// Create web driver instance

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

     // App URL

     String appUrl = "http://www.google.com";

     // Get Selenium instance

Selenium selenium = **new** WebDriverBackedSelenium(driver, appUrl);

     // Tests using selenium

     selenium.open(appURL);

     selenium.type("name=q", "testData");

     selenium.click("name=btnG");

     // Get back the WebDriver instance

WebDriver driverInstance = ((WebDriverBackedSelenium) selenium).getUnderlyingWebDriver();

**29**.I had support of lots of browsers while using Selenium 1.0 and it seems lacking with Selenium 2.0, for example how do I use < awesome> browser while using Selenium 2.0?

There is a class called Capabilities which lets you inject new Capabilities in WebDriver. This class can be used to set testing browser as Safari –

//Instantiate Capabilities

     Capabilities capabilities = **new** DesiredCapabilities()

     //Set browser name

     capabilities.setBrowserName("this awesome browser");

     //Get your browser execution capabilities

CommandExecutor executor = **new** SeleneseCommandExecutor("http:localhost:4444/", "http://www.google.com/", capabilities);

     //Setup driver instance with desired Capabilities

WebDriver driver = **new** RemoteWebDriver(executor, capabilities);

**30**.Are there any limitations while injecting capabilities in WebDriver to perform tests on a browser which is not supported by WebDriver?

Major limitation of injecting Capabilities is that “fundElement” command may not work as expected. This is because WebDriver uses Selenium Core to make “Capability injection” work which is limited by java script security policies.

**31**.Can I change User-Agent while using FF browser? I want to execute my tests with a specific User-Agent setting.

You can create FF profile and add additional Preferences to it. Then this profile could be passed to Firefox driver while creating instance of Firefox –

FirefoxProfile profile = **new** FirefoxProfile();

profile.addAdditionalPreference("general.useragent.override", "User Agent String");

     WebDriver driver = **new** FirefoxDriver(profile);

**32**.Is there any difference in XPath implementation in different WebDriver implementations?

Since not all browsers (like IE) have support for native XPath, WebDriver provides its own implementation for XPath for such browsers. In case of HTMLUnitDriver and IEDriver, html tags and attributes names are considered lower cased while in case of FF driver they are considered case in-sensitive.

**33**.My application uses ajax highly and my tests are suffering from time outs while using Selenium 2.0L.

You can state WebDriver to implicitly wait for presence of Element if they are not available instantly.  By default this setting is set to 0. Once set, this value stays till the life span of WebDriver object. Following example would wait for 60 seconds before throwing ElementNotFound exception –

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

driver.manage().timeouts().implicitlyWait(60, TimeUnit.*SECONDS*);

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

**34**.What if I don’t want to use implicit wait and want to wait only for presence of certain elements?

You can use explicit wait in this situation to wait for presence of certain element before continuing with test execution. You can use “WebDriverWait” and “ExpectedCondition” to achieve this –

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

WebElement myDynamicElement = (**new** WebDriverWait(driver, 60)).until(**new** ExpectedCondition<WebElement>(){

@Override

**public** WebElement apply(WebDriver d) {

**return** d.findElement(By.id("myDynamicElement"));

     }});

This is going to wait up to 60 seconds before throwing ElementNotFound exception.

**35**.What is RemoteWebDriver? When would I have to use it?

RemoteWebDriver is needed when you want to use HTMLUnitDriver. Since HTMLUnitDriver runs in memory, you would not see a browser getting launched –

        // Create HTMLUnitDriver instance

        WebDriver driver = **new** HtmlUnitDriver();

        // Launch Yahoo.com

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

**36**.What all languages available to be used with WebDriver?

Java and C# are on the forefront of WebDriver languages. Support is also available for Python and Ruby. There is also one java script library available for Friefox.

**37**.How do I handle java script alert using WebDriver?

WebDriver would support handling js alerts using Alert interface.

                 // Bring control on already opened alert

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

          // Get the text of the alert or prompt

          alert.getText();

// Click ok on alert

          alert.accept();

**38**.Could I safely execute multiple instances of WebDriver implementations?

As far as HTMLUnitDriver and FF drivers are concerned, each instance would be independent of other. In case of IE driver there could be only one instance of IE driver running on Windows. If you want to execute more than one instance of IE driver then you should consider using RemoteWebDriver and virtual machines.

**39**.Is it possible to interact with hidden elements using WebDriver?

Since WebDriver tries to exercise browser as closely as real users would, hence simple answer is No, But you can use java script execution capabilities to interact with hidden elements.

**40**.I have all my tests written in Selenium 1.0 (Selenium RC), why should I migrate to Selenium 2.0 (WebDriver)?

Because –

·         WebDriver has more compact and object oriented API than Selenium 1.0

·         WebDriver simulates user behaviour more closely than Selenium 1.0, for example if a text box is disabled WebDriver would not be able to type text in it while Selenium 1.0 would be

·         WebDriver is supported by Browser vendor themselves i.e. FF, Opera, Chrome etc

**41**.My XPath and CSS locators don’t always work with Selenium 2.0, but they used to with Selenium 1.0L.

In case of XPath, it is because WebDriver uses native browser methods unless it is not available. And this cause complex XPath to be broken. In case of Selenium 1.0 css selectors are implemented using Sizzle Library and not all the capabilities like “contains” are available to be used with Selenium 2.0

**42**.How do I execute Java Script in Selenium 2.0?

You need to use JavaScriptExecutor to execute java script in Selenium 2.0, For example if you want to find tag name of an element using Selenium 2.0 then you can execute java script as following –

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

String name = (String) ((JavascriptExecutor) driver).executeScript(

     "return arguments[0].tagName", element);

**43**.Why does not my java script execution return any value?

This might happen when you forget to add “return“ keyword while executing java script. Notice the “return” keyword in following statement –

((JavascriptExecutor) driver).executeScript("***return*** window.title;");

**44**.Are there any limitations from operating systems while using WebDriver?

While HTMLUnitDriver, FF Driver and Chrome Driver could be used on all operating systems, IE Driver could be used only with Windows.

**45**.Give me architectural overview of WebDriver.

WebDriver tries to simulate real user interaction as much as possible. This is the reason why WebDriver does not have “fireEvent” method and “getText” returns the text as a real user would see it. WebDriver implementation for a browser is driven by the language which is best to driver it. In case of FF best fit languages are Javascript in an XPCOM component and in IE it is C++ using IE automation.  Now the implementation which is available to user is a thin wrapper around the implementation and user need not know about implementation.

**46**.What is Remote WebDriver Server?

Remote WebDriver Server has two components – client and server. Client is WebDriver while Server is java servlet. Once you have downloaded selenium-server-standalone-.jar file you can start it from command line as –

        java -jar selenium-server-standalone-<version-number>.jar

**47**.Is there a way to start Remote WebDriver Server from my code?

First add Remote WebDriver jar in your class path. You also need another server called “Jetty” to use it. You can start sever as following –

                        WebAppContext context = **new** WebAppContext();

         context.setContextPath("");

         context.setWar(**new** File("."));

         server.addHandler(context);

         context.addServlet(DriverServlet.**class**, "/wd/\*");

SelectChannelConnector connector = **new** SelectChannelConnector();

         connector.setPort(3001);

         server.addConnector(connector);

         server.start();

**48**.But what are the advantages of using Remote WebDriver over WebDriver?

You can use Remote WebDriver when –

·         When you want to execute tests on a browser not available to you locally

·         Introduction to extra latency to tests

But there is one disadvantage of using Remote WebDriver that you would need external servlet container.

**49**.Can you show me code example of using Remote WebDriver?

// Any driver could be used for test

DesiredCapabilities capabilities = **new** DesiredCapabilities();

          // Enable javascript support

          capabilities.setJavascriptEnabled(**true**);

          // Get driver handle

          WebDriver driver = **new** RemoteWebDriver(capabilities);

          // Launch the app

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

**50.**What are the modes of Remote WebDriver

Remote WebDriver has two modes of operations –

*Client Mode:*This is where language bindings connect to remote instance. FF drive and RemoteWebDriver clients work this way.

*Server Mode:*In this mode language bindings set up the server. ChromeDriver works this way.

**51**.What Design Patterns could be used while using Selenium 2.0?

These three Design Patterns are very popular while writing Selenium 2.0 tests –

1.      Page Objects – which abstracts UI of web page

2.      Domain Specific Language – which tries to write tests which could be understood by a normal user having no technical knowledge

3.      Bot Style Tests – it follows “command-like” test scripting

**52**.So do I need to follow these Design patterns while writing my tests?

Not at all, these Design Patterns are considered best practices and you can write you tests without following any of those Design Patterns, or you may follow a Design Pattern which suites your needs most.

**53**.Is there a way to enable java script while using HTMLUnitDriver?

Use this –

HtmlUnitDriver driver = **new** HtmlUnitDriver();

     driver.setJavascriptEnabled(**true**);

or this –

     HtmlUnitDriver driver = **new** HtmlUnitDriver(**true**);

**54**.Is it possible to emulate a browser with HTMLUnitDriver?

You can emulate browser while using HTMLUnitDriver but it is not recommended as applications are coded irrespective of browser you use. You could emulate Firefox 3 browser with HTMLUnitDriver as –

HtmlUnitDriver driver = **new** HtmlUnitDriver(BrowserVersion.FIREFOX\_3);

Or you can inject desired capabilities while instantiating HTMLUnitDriver as –

HtmlUnitDriver driver = new HtmlUnitDriver(capabilities);

**55**.How do I use iPhone Driver?

You should start iPhone SDK and build iPhone driver. Down load iPhone development tools and provision profile. Now iPhone driver can connect through HTTP to the iphone simulator. You can also run simulator on another machine in your network and WebDriver could connect to it remotely.

**56**.Is it possible to convert Selenium IDE test to WebDriver test?

Yes WebDriver style of code can be generated from Selenium IDE

**57**.Can WebDriver handle UntrustedSSLCertificates?

This feature is currently supported in Firefox browser and is awaiting implementation in IE and Chrome drivers.

**58**.Can I carry out multiple operations at once while using WebDriver?

You can use Builder pattern to achieve this. For example if you want to move an element from one place to another you can use this –

          Actions builder = **new** Actions(driver);

          Action dragAndDrop = builder.clickAndHold(element)

                 .moveToElement(otherElement)

                 .release(otherElement)

                 .build();

          dragAndDrop.perform();

**59**.How do I simulate keyboard keys using WebDriver?

There is a KeyBoard interface which has three methods to support keyboard interaction –

* sendKeys(CharSequence)- Sends character sequence
* pressKey(Keys keyToPress) - Sends a key press without releasing it.
* releaseKey(Keys keyToRelease) - Releases a modifier key

**60**.What about Mouse Interaction?

Mouse interface lets you carry out following operations –

* click(WebElement element) – Clicks an element
* doubleClick(WebElement element) - Double-clicks an element.
* void mouseDown(WebElement element) - Holds down the left mouse button on an element.
* mouseUp(WebElement element) - Releases the mouse button on an element.
* mouseMove(WebElement element) - Moves element form current location to another element.
* contextClick(WebElement element) - Performs a context-click (right click) on an element.

**61**.How does Android Webdriver works?

Android WebDriver uses Remote WebDriver. Client Side is test code and Server side is application installed on android emulator or actual device. Here client and server communicate using JSON wire protocol consisting of Rest requests.

**62**.What are the advantages of using Android WebDriver?

Android web driver runs on Android browser which is best real user interaction. It also uses native touch events to emulated user interaction.

But there are some drawbacks also like, it is slower than headless WebKit driver. XPath is not natively supported in Android web view.

**63**.Is there a built-in DSL (domain specific language) support available in WebDriver?

There is not, but you can easily build your own DSL, for example instead of using –

webDriver.findElement(By.*name*("q")).sendKeys("Test");

You can create a more composite method and use it –

**public** **static** **void** findElementAndType(WebDriver webDriver, String elementLocator, String testData) {

webDriver.findElement(By.*name*(elementLocator)).sendKeys(testData);

     }

And now you just need to call method findElementAndType to do type operation.

**64**.What is grid2?

Grid2 is Selenium grid for Selenium 1 as well as WebDriver, This allows to –

·         Execute tests on parallel on different machines

·         Managing multiple environments from one point

**65**.How do I start hub and slaves machines in grid 2?

Navigate to you selenium server standalone jar download and execute following command –

java -jar selenium-server-standalone-.jar -role hub

And you start Slave machine by executing following command –

Java –jar selenium-server-.jar –role webdriver  -hub http://localhost:4444/grid/register -port 6666