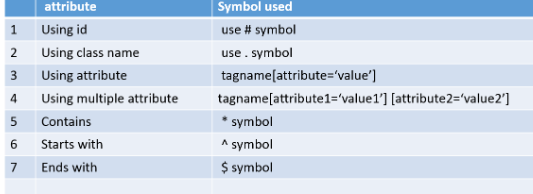
## Symbol used while writing CSS selector in Selenium Webdriver



<html>

<body>

<div>

<input type="text" name="user\_key" id="userid">

<input type="text" name="pass\_key">

</div>

<div>

<input type="text" name="user\_key" id="userid">

<input type="text" name="pass\_key">

</div>

</body>

</html>

**Get div 2 pass key**

body > div:nth-child(2) > input:nth-child(2)

div:nth-child(2) > input[name='pass\_key']

**id\* means contains like**

div:nth-child(2) > input[name\*='pass']

**^ starts with**

div:nth-child(2) > input[name^='pass']

**$ Ends with**

div:nth-child(2) > input[name$='key']

if id is like <input type="text" name="pass\_key" id="/pass\_word">

$("#\\/pass\_word")

**Issue on IE in sleenium**

**Issues** 1-**openqa.selenium.NoSuchWindowException**

This is a common issue with Selenium and you can avoid this by doing some IE setting, which we are going to discuss now.

**Issue 2**-**sendKeys works very slow it takes 1-2 second to type each character.**

This is a known issue with Selenium and it only happens once you work with IE 64 bit driver.

Solution- You can download IE Driver 32 bit and start using it, even you are working with 64 bit OS this 32 bit IE driver works every time.

**Issue 3-Unexpected error launching Internet Explorer. Protected Mode must be set to the same value**

**Issue 4-Unexpected error launching Internet Explorer. Browser zoom level was set to 0%**

**Handle Untrusted certificate in IE**

Click on the link using driver.navigate() method with[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)as below :-

driver.navigate ().to ("javascript:document.getElementById('overridelink').click()");

1. The second method is quite similar to chrome SSL Handling code

DesiredCapabilities capabilities = new DesiredCapabilities();

capabilities.setCapability(CapabilityType.ACCEPT\_SSL\_CERTS, true);

 Difference between findElement() and findElements() method:

1. Return type of findElement() is a a web element while return type of findElements() is a List<WebElement>.
2. findElement() method **will throw noSuchElementException if web element is not found while findElement() will not throw any exception. It will return an empty List<WebElement>.**

# What is a framework?

# A framework defines a set of rules or best practices which we can follow in a systematic way to achieve the desired results.

# Linear Scripting Framework:

# Linear Scripting Framework is a basic level test automation framework which is in the form of ‘Record and Playback’ in a linear fashion. This framework is also known as ‘Record and Playback’ framework. This type of framework is used to test small sized applications.

# Data-driven Framework:

# Data driven test automation framework is focused on separating the test scripts logic and the test data from each other. Allows us to create test automation scripts by passing different sets of test data. The test data set is kept in the external files or resources such as MS Excel Sheets, MS Access Tables, SQL Database, XML files etc.,

# Keyword Driven Testing Framework:

# It is also known as table-driven testing or action word based testing. In Keyword-driven testing, we use a table Format to define keywords or action words for each function or method that we would execute. It performs Automation test scripts based on the keywords specified in the excel sheet. By using this Framework, testers can work

# Hybrid Driven Testing Framework: Hybrid Test automation framework is the combination of two or more frameworks mentioned above. It attempts to leverage the strengths and benefits of other frameworks for the particular test environment it manages. Most of the teams are building this hybrid driven framework in the current market.

# BDD:

# Behavior-Driven Development addresses this problem by showing you how to test. You should not test Implementation, but instead behavior.

# Conclusion

# Unit Testing gives you the what. Test-Driven Development gives you the when. Behavior Driven-Development gives you the how. Although you can use each individually, you should combine them for best results as they complement each other very nicely.

# DD vs. BDD

# BDD is in a more readable format by every stake holder since it is in English, unlike TDD test cases written in programming languages such as Ruby, Java etc.

# BDD explains the behavior of an application for the end user while TDD focuses on how functionality is implemented. Changes on functionality can be accommodated with less impact in BDD as opposed to TDD.

# BDD enables all the stakeholders to be on the same page with requirements which makes acceptance easy, as opposed to TDD.

# These tests are helpful, but only helpful to one group of people: engineers. BDD is useful for communicating with every member of a cross-functional product team.

# Where did you use absract classes in Selenium

# In locators

# Where did you use Interface classes in Selenium

# Like we use webDriver

# Where did you use PolyMorphism in Selenium

# When i have to select dropdwon one mthod locator with Tesxt and in ANothe locator with index

# How to Print error log in extent report

// new instance

ExtentReports extent = new ExtentReports(file-path, replaceExisting);

// starting test

ExtentTest test = extent.startTest("Test Name", "Sample description");

// step log

test.log(LogStatus.INFO, "Click on the object");

Reference:

try{

int num[]={1,2,3,4};

System.out.println(num[5]);

}catch(Exception e){

test.log(LogStatus.INFO/ERROR, ExceptionUtils.getStackTrace(e));

}

# Exception Handling in Selenium

Your Selenium test should be able to fail, but not because of exceptions that are thrown. If your test is failing from exceptions then quite likely you have no exception handling. By doing this, you don’t have the opportunity to cleanup the WebDriver object at the end of the test.

The tests should be failing under your terms only for example, you should never be getting exceptions like NullPointerException but if you are getting such as ElementNotFoundException, then also it is good idea to catch the exception, stop the further execution and end your test in a Logical way.

**Example 1:** I do not use any Page Object Factory but I use my own Page Object Pattern and I always print error logs and take screenshot on any exception I encounter. Please look at the code below:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | public static WebElement btn\_ReportCategory(WebDriver driver) throws Exception{            try{                WebElement element = driver.findElement(By.linkText("+ Report Categories"));            }catch (Exception e){    // Printing logs for my report                Log.error("Report Category button element is not found.");    // Taking screenshot for defect reporting    Utils.captureScreenShot();    // After doing my work, now i want to stop my test case                throw(e);            }    // This will return the Element in case of no Exception            return element;        } |

**Example 2:**TimeoutException using Selenium WebDriver.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | try{        myTestDriver.findElement(By.xpath("//\*[@id='register']")).click();    }catch (TimeoutException toe) {    wait.until( ExpectedConditions.elementToBeClickable(By.xpath("//\*[@id='register']")));    myTestDriver.findElement(By.xpath("//\*[@id='register']")).click();    }catch (Exception e) {    Log.error("Register element is not found.");    throw(e);        }    } |

**Example 3:**Let’s assume that in Selenium WebDriver you want to verify the presence of any element on the page. You would not be able to get this with element locator because if the element is present, your locator will work and you will easily be able to print that the element is present but in case your element is not present on the page, your locator will fail and simply throw the exception. This case would be easily handled with the self-written function.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public static boolean verifyObjectPresent(WebDriver driver) {        try {         driver.findElement(By.linkText("+ Report Categories"));         return true;        } catch (Exception e) {         return false;        }    } |

# Reportng Reporting needs 3 jar

1.reportng

2.velocity

3.Guice google jar

# 

# Handle untrusted certificate using Selenium in Firefox Chrome IE

https://www.youtube.com/watch?v=7q8viGgiVSc

http://learn-automation.com/handle-untrusted-certificate-selenium/

By default selenium dost not allow any untrusted certificate





Handle untrusted certificate in Firefox

Now let’s see with threadPoolSize implementation



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | public class InvocationCnt {      @Test(threadPoolSize=3, invocationCount=5)    public void test1() throws InterruptedException    {          System.out.println("Thread ID: "+Thread.currentThread().getId());    }  } |

*OutPut:*  
*[ThreadUtil] Starting executor timeOut:0ms workers:5 threadPoolSize:3*  
*Thread ID: 11  
Thread ID: 12  
Thread ID: 13*  
*Thread ID: 12*  
*Thread ID: 11*  
Above method ran 5 times, but 3 threads shared the 5 times of test run…(threads 11, 12 and 13)

Now let’s see threadPoolSize with timeOut implementation

**invocationcount**- no of times method has to execute

**thread-count**- no of threads where all methods has to be shared



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public class InvocationCnt {      @Test(threadPoolSize=3, invocationCount=5, timeOut=2000)    public void test1() throws InterruptedException    {       Thread.sleep(1500);       System.out.println("Thread ID: "+Thread.currentThread().getId());    }  } |

OutPut:  
*[ThreadUtil] Starting executor timeOut:2000ms workers:5 threadPoolSize:3*  
*Thread ID: 11*  
*Thread ID: 12*  
*Thread ID: 13*

**HUb and node setup**

C:\Users\Ram.Sahu\Downloads>java -jar selenium-server-standalone-3.14.0.jar -role hub

ram.sahu>java -Dwebdriver.chrome.driver="Downloads/Softwares and Projects/chromedriver\_34.exe" -jar selenium-server-standalone-3.14.0.jar -role node -hub <http://192.168.29.110:4444/grid/register>

java -Dwebdriver.chrome.driver="Downloads/Softwares and Projects/chromedriver\_34.exe" -jar selenium-server-standalone-3.14.0.jar -role node -hub [http://10.50.32.33:4444/grid/register](http://192.168.29.110:4444/grid/register)

cap.setBrowserName("chrome");

cap.setPlatform(Platform.***WINDOWS***);

ChromeOptions option = **new** ChromeOptions();

option.addArguments("--disable-infobars");

option.addArguments("start-maximized");

option.merge(cap);

String huburl="http://192.168.29.110:4444/wd/hub";

WebDriver driver = **new** RemoteWebDriver(**new** URL(huburl),option);

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

driver.quit();

**What is the difference between Desiredcapabilities & Chromeoptions and when to use them?**

Capabilities are options that you can use to customize and configure a ChromeDriver session. This page documents all ChromeDriver supported capabilities and how to use them.

<http://chromedriver.chromium.org/capabilities>

1. Use the ChromeOptions class. This is supported by Java, Python, etc.
2. Use the DesiredCapabilities class. This is supported by Python, Ruby, etc. While it is also available in Java, its usage in Java is deprecated.

## Using the ChromeOptions class

You can create an instance of ChromeOptions, which has convenient methods for setting ChromeDriver-specific capabilities. You can then pass the ChromeOptions object into the ChromeDriver constructor:

ChromeOptions options = new ChromeOptions();

options.addExtensions(new File("/path/to/extension.crx"));

ChromeDriver driver = new ChromeDriver(options);

Since Selenium version 3.6.0, the ChromeOptions class in Java also implements the Capabilitiesinterface, allowing you to specify other WebDriver capabilities not specific to ChromeDriver.

ChromeOptions options = **new** ChromeOptions();

// Add the WebDriver proxy capability.

Proxy proxy = **new** Proxy();

proxy.setHttpProxy("myhttpproxy:3337");

options.setCapability("proxy", proxy);

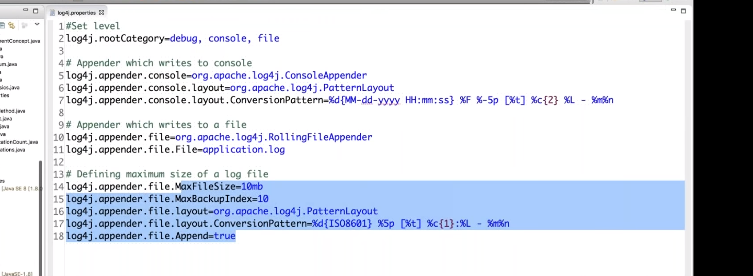
// Add a ChromeDriver-specific capability.

options.addExtensions(**new** File("/path/to/extension.crx"));

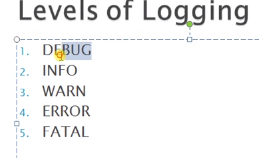
ChromeDriver driver = **new** ChromeDriver(options);

**HeadlessFirefox**

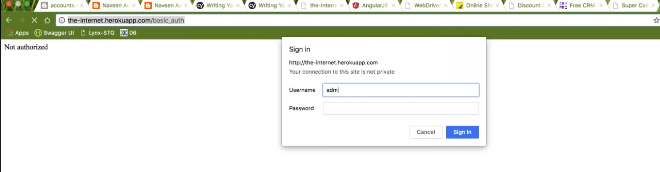
****

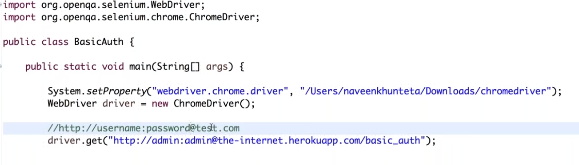
****

**https://www.youtube.com/watch?v=0XowYwfvbo8&index=62&list=PLFGoYjJG\_fqo4oVsa6l\_V-\_7-tzBnlulT**

****

**Basic authentication**

****

****

**Page Facotry**

Selenium Page Factory Pattern is like an extension to [Page Object Model](http://www.seleniumeasy.com/selenium-tutorials/page-object-model-framework-introduction), but Page Factory is much enhanced model. To start with, we just need to import package ‘org.openqa.selenium.support.PageFactory’

"Factory class can be used to make using Page Objects simpler and easier".

We use [Page Factory pattern](https://selenium.googlecode.com/git/docs/api/java/org/openqa/selenium/support/PageFactory.html) to initialize web elements which are defined in Page Objects.

We should initialize page objects using initElements () method from PageFactory Class as below, Once we call initElements () method, all elements will get initialized. PageFactory.initElements() static method takes the driver instance of the given class and the class type, and returns a Page Object with its fields fully initialized.

Home homePage = **new** HomePage(driver);

PageFactory.initElements(driver, homePage);

Or,

*// To initialize elements.*

HomePage homePage = PageFactory.initElements(driver, HomePage.**class**);

Or, **as a constructor for page class as below:**

**public** **HompePage**(WebDriver driver) {

**this**.driver = driver;

PageFactory.initElements(driver, **this**);

}

**// Check whole Page is loaded completely**

**public void checkPageIsReady() {**

**JavascriptExecutor js = (JavascriptExecutor)driver;**

**for (int i=0; i<25; i++){**

**try {**

**Thread.sleep(1000);**

**}catch (InterruptedException e) {}**

**if (js.executeScript("return document.readyState").toString().equals("complete")){**

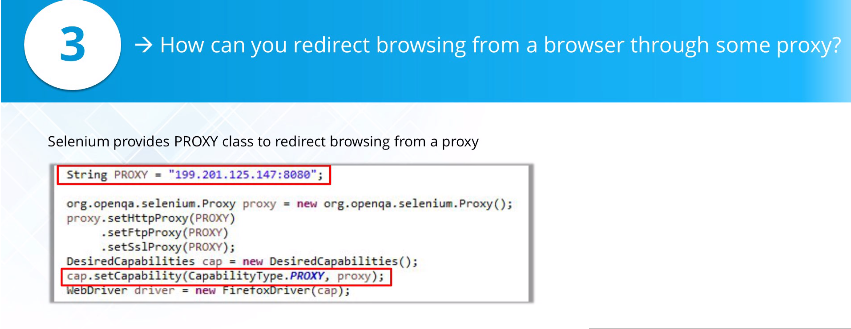
**break;**

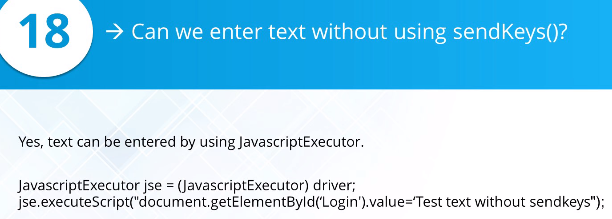
**}**

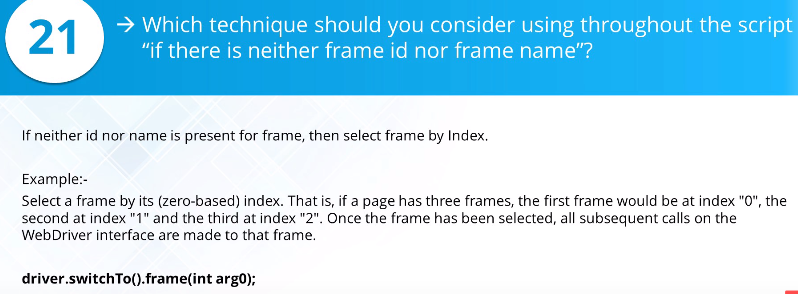
**}**

**}**

**}**

****

****

****

**SEt IE capability**

DesiredCapabilities capabilities = DesiredCapabilities.internetExplorer();

capabilities.setCapability(CapabilityType.BROWSER\_NAME, "IE");

capabilities.setCapability(InternetExplorerDriver.

INTRODUCE\_FLAKINESS\_BY\_IGNORING\_SECURITY\_DOMAINS,true);

System.setProperty("webdriver.ie.driver", "C:\\IEDriverServer.exe");

//it is used to initialize the IE driver

**Load Firefox Profile**

ProfilesIni profile = new ProfilesIni();

/FirefoxProfile myprofile = profile.getProfile("xyzProfile");

WebDriver driver = new FirefoxDriver(myprofile)

**How to define Sepcific download path in chrome and Firefox**

String downloadFilepath = "/path/to/download";

HashMap<String, Object> chromePrefs = new HashMap<String, Object>();

chromePrefs.put("profile.default\_content\_settings.popups", 0);

chromePrefs.put("download.default\_directory", downloadFilepath);

ChromeOptions options = new ChromeOptions();

options.setExperimentalOption("prefs", chromePrefs);

WebDriver driver = new ChromeDriver(options);

/Download setting

FirefoxProfile profile = new FirefoxProfile();

profile.setPreference("browser.download.folderlist", 2);

profile.setPreference("browser.helperapps.neverAsk.saveToDisk","jpeg");

profile.setPreference("browser.download.dir", "C:\\Users\\Admin\\Desktop\\ScreenShot\\pic.jpeg");

driver = new FirefoxDriver(profile);

**Print array in Java-**---->System.***out***.println((Arrays.*toString*(*arr*)));

Print List in JAVA-------------------> Arrays.toString(list.toArray())

2d array -System.out.println(Arrays.deepToString(array));

ContextClick or Right click

Actions action = new Actions(driver);

action.contextClick(WebElement).build().perform();

Double Click

Actions action = new Actions(driver);

WebElement element=driver.findElement(By.linkText("TEST"));

action.doubleClick(element).perform();

**Coridinate click**

Point point = element.getLocation();

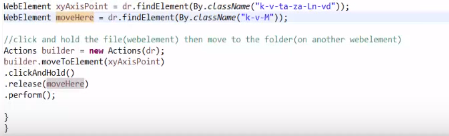
int xcord = point.getX();

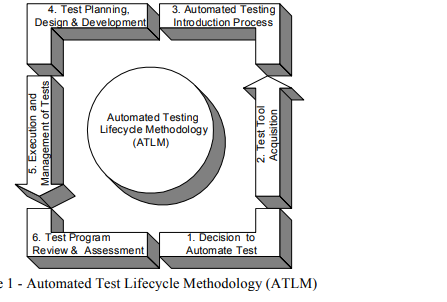
int ycord = point.getY();

Actions action = new Actions(driver);

action.moveToElement(element, xcord, ycord).click().build().perform();

Move to file to a folder---- MovetoElement



****

**How Test Data can be Generated -**

* Manually
* Mass copy of data from production to testing environment
* Mass copy of test data from legacy client systems
* Automated Test Data Generation Tools

**Test Data for Black Box Testing**

In Black Box Testing the code is not visible to the tester . Your functional test cases can have test data meeting following criteria -

* **No data**: Check system response when no data is submitted
* **Valid data**: Check system response when Valid  test data is submitted
* **Invalid data**: Check system response when *InValid*  test data is submitted
* **Illegal data format**: Check system response when test data is in invalid format
* **Boundary Condition Data set**: Test data meeting boundary value conditions
* **Equivalence Partition Data Set**: Test data qualifying your equivalence partitions.
* **Decision Table Data Set**: Test data qualifying your decision table testing strategy
* **State Transition Test Data Set**: Test data meeting your state transition testing strategy
* **Use Case Test Data**: Test Data in-sync with your use cases.

**Q #4) Can you achieve 100% automation?**

100% automation would be difficult to achieve because there would be many edge test cases and some cases which are executed seldom. Automating these cases which are not executed that often will not add value to the automated suite.

**Q #5) Currently I do not have any automation in place in my project, now I want to implement automation, what would be my steps?**

* First, identify which type of testing/test cases you want to automate
* Identify the tool
* Design the framework
* Create the utility files and environment files
* Start scripting
* Identify and work on the reporting
* Allocating time for enhancing and maintaining the scripts.

### ****Test Automation Tool Evaluation Criteria****

**1)** Do you have the necessary skilled resource to allocate for automation tasks?

**2)** What is your budget?

**3)** Does the tool satisfy your testing needs? Is it suitable for the project environment and technology you are using? Does it support all tools and objects used in the code? Sometime you may get stuck for small tests due to inabilities of the tool to identify the objects used in the application.

*I consider above three factors as most important for selecting any tool.*

**4)** Does the tool provide you the free trial version so that you can evaluate it before making a decision? Also, does the tool have all features available in the trial version?

**6)** How is the tool learning curve? Is the learning time acceptable for your goals?

**7)** Do you want automation tool for only your project needs or you are looking for a common tool for all projects in your company? It would be a good choice if you select a tool that supports most of the coding languages on your projects.

**8 )** Which testing types does it support? A tool which supports maximum testing types (Unit, functional, regression etc.) is always a better choice. Warning – Don’t go for a tool just because it is supporting all testing types. It’s also important that the tool should be powerful enough to automate your complex requirements.

**9)** Does the tool support easy interface to create and maintain test scripts? Record and playback tool with abilities to edit recorded scripts could be a good solution.

**10)**  Does it provide simple interface yet powerful features to accomplish complex tasks?

**12)**  Does it provide the powerful reporting with graphical interface? Clear and concise reports will always help you to conclude the test results quickly.

**13)**  Does it integrate well with your other testing tools like project planning and [test management tools](https://www.softwaretestinghelp.com/software-test-case-management-tools/)?

**You may also want to consider other criteria like:**

**14)**  Tool vendor refund policy

**15)** Existing customer reviews for the tool

**16)**  Is the vendor providing initial training?

**Q #8) What is a framework?**

A framework is a set of a structure of the entire automation suit. It is also a guideline if followed can result in a structure which is easy to maintain and enhance. These guidelines include:

Frameworks are guidelines and not mandatory rules, so we can do without a framework, but if we create it and follow it, enhancing and maintaining would be easy to implement.

**Coding standards**

* Handling the test data
* Maintaining and handling the elements (object repository in QTP)
* Handling of environment files and properties file
* Reporting of data
* Handling logs

**Q #9) What are the attributes of a good framework?**

The characteristics are:

* **Modular** – The framework should be adaptable to change. Testers should be able to modify the scripts as per the environment or login information change
* **Reusable** – The commonly used methods or utilities should be written in a common file which is accessible to all the scripts.
* **Consistent** – The suite should be written in a consistent format by following all the accepted coding practices.
* **Independent** – The scripts should be written in such a way that they are independent of each other. In case one test fails, it should not hold back remaining test cases (unless it is a login page)
* **Logger** – It is good to have implemented the logging feature in the framework. This would help in case our scripts run for longer hours (say nightly mode), if the script fails at any point of time, having the log file will help us to detect the location and the type of error.
* **Reporting** – It is good to have reporting feature automatically embedded into the framework. Once the scripting is done, we can have the results and reports sent via an email.
* **Integration** – Automation framework should be such that it is easy to integrate it with other application like continuous integration or triggering the automated script as soon as the build is deployed.

**4) What are the steps involved in the Automation Process?**

* Selecting the Test tool
* Select a framework
* Define scope of automation
* Planning, design, and development
* Test execution
* Maintenance

**5) What are the points that are covered while planning phase of automation?**

During planning phase of automation things which must be taken in concern are

* Selection the "right" Automation tool
* Selection Automation Framework if any
* List of In scope and out of scope items for automation
* Test Environment Setup
* Preparing Grant Chart of Project timelines for test script development & execution.
* Identify Test Deliverables

**6) In what condition we cannot use automation testing for the Agile method?**

Automation testing is not useful for agile methods in following conditions

* When Agile testing always ask for changes in requirements
* When Exhaustive level of documentation is required in Agile
* Only suitable for those regression tests during agile testing like continuous integration

**7) What are the primary features of good automation tool?**

* Test Environment support and easy to use
* Good debugging facility
* Robust object identification
* Object and Image testing abilities
* Object identification
* Testing of database
* Support multiple frameworks

**9) What is the scripting standard while performing automation testing?**

While writing the scripts for automation, you must consider following things,

• Uniform naming convention.

• 3 Lines of comments for every 10 lines of code

• Adequate indentation.

• Robust error handling and recovery scenario

• Use of Frameworks wherever possible

**11) On what basis you can map the success of automation testing?**

* Defect Detection Ratio
* Automation execution time and time savings to release the product
* Reduction in Labour & other costs

**Framewrok vs Architecture**

Simply put -- architecture is theory, framework is implementation.

* A **Framework** consists of one or more libraries, but the difference is that *Inversion of Control*applies. The application registers with the framework (often by implementing one or more interfaces), and the framework calls into the application, which may call back into the framework. A framework often exists to address a particular general-purpose Domain (such as web applications, or workflows, etc.). like **TestNG**
* **Architecture** consists of the guiding principles behind a given application. It is not strongly tied to a particular framework or library.**like - Page object Model**

**Cons of automating a test at UI level**

**Not reliable:**

**1.** Automated checks can break because a minor UI change was implemented, or a service is down or there are network issues which are not relevant to the application under test but could impact the automated checks.

**Maintenance Time and Effort-** You have to accept the fact that if you have automated checks in place, you will need to spend the time upgrading the relevant tests as the application is upgraded. If the regression packs are not kept up to date, you start seeing failing tests that fail due to upgrades rather than identifying real bugs, as well as tests that are no longer applicable.

**Slow feedback-** When a functionality is ready to test, sometimes it is quicker to do a manual check rather than waiting first to automate the test and then run the test and check the results.

**Not many bugs found**

Majority of the bugs seem to be found by “accident” or when performing exploratory testing. This is probably due to the fact that during each exploratory testing session we could be testing the application in different ways thus finding new loopholes through the application.

Automated regression checks on the other hand always follow a given path and sometimes with the same set of test data. This reduces the chance of finding new defects in the

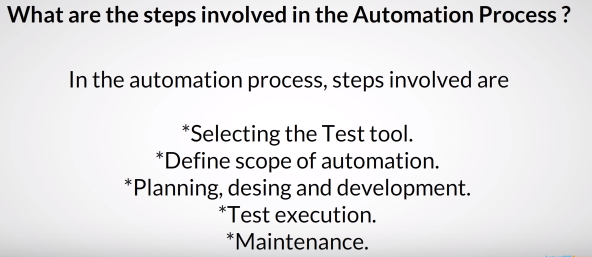
Pros of Automations:

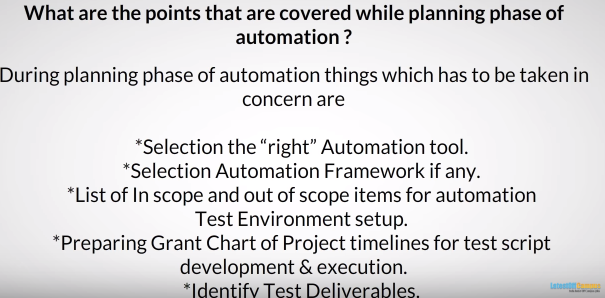
* **Cross-Platform Testing**
* **Image Comparison Techniques**
* **UI Validation Techniques**
* **Functional Testing**
* **Report Generation**
* **Integration with Defect Tracking Tools**
* **Continuous Integration**
* **Reduces Time Efforts**

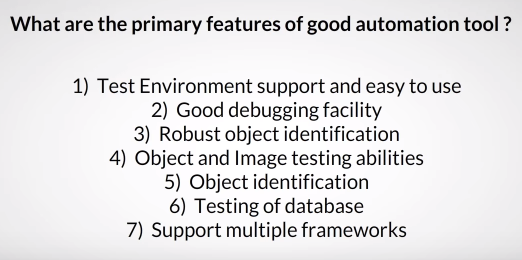
**3) When will you not automate testing?**

One should not automate in following cases

* When the Application Under Test changes frequently
* One time test cases
* Adhoc – Random testing
* usabilty testing test cases





****

**driver.get() :** It's used to go to the particular website , But it doesn't maintain the browser History and cookies so , we can't use forward and backward button , if we click on that , page will not get schedule

**driver.navigate() :** it's used to go to the particular website , but it maintains the browser history and cookies, so we can use forward and backward button to navigate between the pages during the coding of Testcase

The build() method is used compile all the listed actions into a single step.  
we have to use build() when we are performing sequence of operations and no need to use only if we are performing single action.

WebDriver protocol to communicate with Gecko browsers, such as Firefox (Version after 47).

# ****New Changes in Selenium 3.0****

Minimum Java version required is 1.8+

* **Firefox is fully supported at version 47.0.1 or earlier. Gecko driver provides support  
  for later versions of Firefox. Firefox 47.0.1 and before would not need GeckoDriver**
* Selenium project will not actively support the WebDriver API.
* **The Selenium RC APIs are only available via the legacy-rc package.**
* Support Firefox browser via Mozilla’s gecko driver.
* **Selenium 3.0 does not have HeadLess Driver (HtmlWebDriver)**
* **Official support for IE browser requires 9 or above version.**
* **No need to maximize the browser using driver.manage().window().maximize(); By default it will maximize the window.**
* Apple has come up with its own SafariDriver to run your tests in Safari on Mac
* Selenium RC core library won’t be supported.
* Selenium WebDriver has become the W3C Standard. The W3C standard will encourage compatibility across different software implementations of WebDriver API.
* All WebDrivers are moved out of selenium and vendors will be responsible for owning the drivers for maintenance, which will be the relief for maintenance team or open source team to get better implementation.

How To Run Selenium Tests In Headless Google Chrome

System.setProperty("webdriver.chrome.driver",

"ChromeDriverPath");

ChromeOptions options = new ChromeOptions();

options.addArguments("headless");

options.addArguments("window-size=1200x600");

WebDriver driver = new ChromeDriver(options);

driver.get("https://contentstack.built.io");

driver.get("https://www.google.co.in/");

System.out.println("title is: " + driver.getTitle());

File scrFile = ((TakesScreenshot) driver)

.getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(scrFile, new File("pathTOSaveFile"));

driver.quit();

}

## ****Issues found in Selenium 3.0****

* Unsupported Marionette protocol version 2, required 3 when our Firefox is 64 bit.
* driver.close() is not working

##### here are multiple ways in which you can re-run failed tests.

1. Use a retry analyzer and implement the retry running of tests using this.

2. In CI systems such as Jenkins, you have the option of adding a post build step wherein you can write up a simple shell script which basically checks if there is a testng-failed.xml file available in the test output path and if it is available then kick off the build [ by running either the ant task or the maven task ] by providing the suite file to be executed as testng-failed.xml

Note:

Approach 2 is going to be messing up your test results (if you are dependent on the default reports that TestNG provides) because it would get overwritten. So you may have to put in additional checks in those areas as well.

* 1. Separate your tests from your test automation framework

**Q #9) what are the attributes/best practices of a good framework?**

* Modular – The framework should be adaptable to change. Testers should be able to modify the scripts as per the environment or login information change
* Reusable – The commonly used methods or utilities should be written in a common file which is accessible to all the scripts.
* Consistent – The suite should be written in a consistent format by following all the accepted coding practices.
* Independent – The scripts should be written in such a way that they are independent of each other. In case one test fails, it should not holdback remaining test cases (unless it is a login page)
* Logger – It is good to have implemented the logging feature in the framework. This would help in case our scripts run for longer hours (say nightly mode), if the script fails at any point of time, having the log file will help us to detect the location and the type of error.
* Reporting – It is good to have reporting feature automatically embedded into the framework. Once the scripting is done, we can have the results and reports sent via an email.
* Integration – Automation framework should be such that it is easy to integrate it with other application like continuous integration or triggering the automated script as soon as the build is deployed.
* Use wait instead of Thread.sleep()
* Take screenshots for failure investigation
* Poratable framework

**Q #15) What are the different types of frameworks?**

Different types of [framework](https://www.softwaretestinghelp.com/test-automation-frameworks-selenium-tutorial-20/)available are:

* Keyword driven framework
* Data Driven framework
* Hybrid Framework
* Linear Scripting

**Q #16) Can you tell some good coding practices while automation?**

Some of the good coding practices include:

* Add appropriate comments
* Identify the reusable methods and write it in separate file
* Follow the language specific coding conventions
* Maintain the test data in a separate file
* Run your scripts regularly

**Q #17) Any kind of test which you thing should not be automated?**

* Test which are seldom executed
* Exploratory testing
* Usability testing

**Q #19) How do you select which automation tool is best suited for you?**

Selecting the automation tool depends upon various factors like:

* Scope of the application which we want to automate
* Management overhead like cost and budget
* Time to learn and implement the tool
* Type of support available for the tool.
* Limitation of the tool

**Handling Dynamic Elements in Selenium**

1.Using Xpath functions ,Sibling, Ancesstors parnet, Stratswith, contains

2. 4. Identify Element by Index

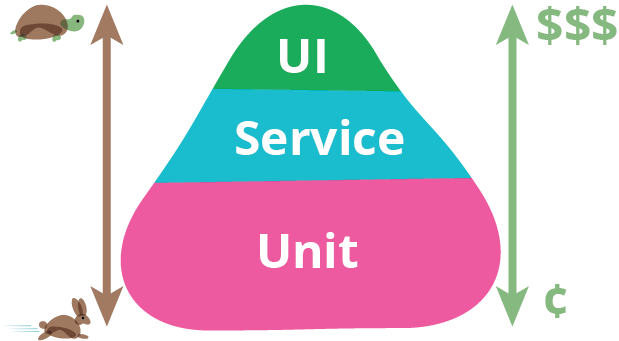
3. Identify Element with reference of a closest stable element

4.Absolute xpath

# [TestPyramid](https://martinfowler.com/bliki/TestPyramid.html)

**The test pyramid is a way of thinking about different kinds of automated tests should be used to create a balanced portfolio.** Its essential point is that you should have many more low-level [UnitTests](https://martinfowler.com/bliki/UnitTest.html) than high level[BroadStackTests](https://martinfowler.com/bliki/BroadStackTest.html) running through a GUI.

For much of my career test automation meant tests that drove an application through its user-interface. Such tools would often provide the facility to record an interaction with the application and then allow you to play back that interaction, checking that the application returned the same results. Such an approach works well initially. It's easy to record tests, and the tests can be recorded by people with no knowledge of programming.

**

Switching to tab

ArrayList<String> tabs2 = new ArrayList<String> (driver.getWindowHandles());

driver.switchTo().window(tabs2.get(1));

driver.close();

driver.switchTo().window(tabs2.get(0));

**Chalenges Faced with Selenium**

1. Image or text overlapping issue
2. **No facility to deal with Captcha**
3. **Doesn’t support any non web based (Like Win 32, Java Applet, Java Swing, .Net Client Server etc) applications**
4. When you compare selenium with QTP, Silk Test, Test Partner and RFT, there are many challenges in terms of maintainability of the test cases
5. **Since Selenium is a freeware tool, there is no direct support if one is in trouble with the support of applications**
6. **Bitmap comparison is not supported by Selenium**
7. **Any reporting related capabilities, you need to depend on third party tools**
8. You need to learn any one of the native language like (.Net, Java, Perl, Python, PHP, Ruby) to work efficiently
9. Difficult to identify dynamic objects
10. JavaScript sandbox, Flash, Applets, Silverlight, and HTML 5’s Canvas all present problems in Selenium

# [How to fill CAPTCHA using Test automation?](https://sqa.stackexchange.com/questions/17022/how-to-fill-captcha-using-test-automation)

**You don't**, that is the whole idea behind a good [CAPTCHA](https://en.wikipedia.org/wiki/CAPTCHA).

CAPTCHA stands for:

Completely Automated Public Turing test to tell Computers and Humans Apart

So, by definition, resolving a CAPTCHA cannot be automated: Otherwise it could not tell computers and humans apart and hence fails being a CAPTCHA.

Howto handle a CAPTCHA in a **test environment**:

1. If you need to test an application which uses CAPTCHA, then you need to get the development team to build in a workaround/backdoor which is only active in the test environment.
2. Use a service like [http://www.deathbycaptcha.com](http://www.deathbycaptcha.com/) which has an API to return the text of the CAPTCHA for you. Average response is 15 seconds with a result rate of 90% or so they say. Sounds like something to try.
3. If your are using custom CAPTCHA module, you can ask developer to generate an API of CAPTCHA generation for testing environment.
4. Either ask dev team for a workaround, like configure CAPTCHA in test environment in such a way it will always accept 1 specific value

Compare Two Images

Depending on how you define the same. If you mean the exact same file, you can do an md5sum of the files. That will be the same for every type of file I guess.

BufferedImage imgA = ImageIO.read(new File("C:/img/picA.jpg"));

BufferedImage imgB = ImageIO.read(new File("C:/img/picB.jpg"));

boolean bufferedImagesEqual(BufferedImage img1, BufferedImage img2)

{

if (img1.getWidth() == img2.getWidth() && img1.getHeight() == img2.getHeight())

{

for (int x = 0; x < img1.getWidth(); x++)

{

for (int y = 0; y < img1.getHeight(); y++)

{

if (img1.getRGB(x, y) != img2.getRGB(x, y))

return false;

}

}

} else {

return false;

}

return true;

}

getRGB(int x, int y) return you the value of color pixel at location **(x,y)**.  
You are misinterpreting the returned value.  
It is in the binary format. like 11...11010101 and that is given to you as int value.  
If you want to get RGB (i.e. Red, Green, Blue) components of that value use Color class. e.g.

Color mycolor = new Color(img.getRGB(x, y));

Then you can get Red, Green, or Blue value by using getRed(), getGreen(), getBlue(), getAlpha(). Then an int value will be returned by these methods in familiar format having value 0 < value < 255

int red = mycolor.getRed();

If you don't wants to use Color class then you will need to use bitwise operations to get its value.

2.You can calculate hash of images and then compare the hashes for equality. [SO](https://stackoverflow.com/questions/304268/getting-a-files-md5-checksum-in-java) has a question on calculating hash

3.    @Test

    public void imageComaparision() throws IOException

    {

        System.setProperty("webdriver.gecko.driver", "D:\\Selenium\\geckodriver.exe");

        driver = new FirefoxDriver();

        driver.get("<http://demo.automationtesting.in/Register.html>");

        public float compareImage(File fileA, File fileB) {

float percentage = 0;

try {

// take buffer data from both image files //

BufferedImage biA = ImageIO.read(fileA);

DataBuffer dbA = biA.getData().getDataBuffer();

int sizeA = dbA.getSize();

BufferedImage biB = ImageIO.read(fileB);

DataBuffer dbB = biB.getData().getDataBuffer();

int sizeB = dbB.getSize();

int count = 0;

// compare data-buffer objects //

if (sizeA == sizeB) {

for (int i = 0; i < sizeA; i++) {

if (dbA.getElem(i) == dbB.getElem(i)) {

count = count + 1;

}

}

percentage = (count \* 100) / sizeA;

} else {

System.out.println("Both the images are not of same size");

}

} catch (Exception e) {

System.out.println("Failed to compare image files ...");

}

return percentage;

}

        driver.quit();

**Q.  In Selenium variables are stored in \_\_\_\_\_\_\_\_\_\_\_\_\_ .**  
*- Published on 03 Aug 15*

**a.** storedVars

**b.** storedVariables

**c.** VariablesStore

**d.** All of the above

**ANSWER: storedVars**

**120. What all things you should consider before selecting automation tools for the AUT?**

* Technical Feasibility
* Complexity level
* Application stability
* Test data
* Application size
* Re-usability of automated scripts
* Execution across environment

**Parent:**

$x("(//input[contains(@name,'sQuery')]/parent::\*)")

**Parents of Parent (Grand Parent): in trivago.com example**

$x("((//input[contains(@name,'sQuery')]/parent::\*)/parent::\*)")

OR

**$x("(//div[contains(@id,'fd4')]/ancestor::\*)[last()-1]")**

**to display all siblings**

**$x("(//div[@id='pagelet\_sidebar'])//preceding-sibling::\*")**

**to display all childs**

**$x("(//div[@id='bluebarRoot'])//following-sibling::div/child::\*")**

**$x("(//div[contains(text(),'THE WORLDS LARGEST WEB DEVELOPER SITE')])//preceding-sibling::\*")**

**$x("(//div[@id='pagelet\_sidebar'])//preceding-sibling::\*")**

**$x("//label[contains(text(),'Query related to')]//following-sibling::div/div[1]")**

**$x("//label[contains(text(),'Query related to')]//following-sibling::div[text(),'test']")**

**$x("//label[contains(text(),'Query related to')]//following-sibling::div/div/child::\*")**

**$x("//input[@id='product']//preceding-sibling::div[1]/div/input")**

**x("//input[@id='product']//preceding-sibling::\*[1]")**

**Get Last Element input**

**$x("//form[@id='contactus-form']/input[last()]")**

**$x("(//div[contains(text(),'test123')])")**

**$x("(//div[text()='test123'])")**

//a[1] # first <a>

//a[last()] # last <a>

//ol/li[2] # second <li>

//ol/li[position()=2] # same as above

//ol/li[position()>1] # :not(:first-child)

|  |  |
| --- | --- |
| **AxisName** | **Result** |
| Ancestor  $x("//input[@id='custom']/ancestor::\*") | Selects all ancestors (parent, grandparent, etc.) of the current node |
| ancestor-or-self  $x("//input[@id='custom']/ancestor::\*") | Selects all ancestors (parent, grandparent, etc.) of the  current node and the current node itself |
| attribute $x("//input[@id='custom']/attribute::\*") $x("//input[@id='custom']/attribute::\*") | Selects all attributes of the current node |
| Child | Selects all children of the current node |
| Descendant | Selects all descendants (children, grandchildren, etc.)  of the current node |
| descendant-or-self | Selects all descendants (children, grandchildren, etc.)  of the current node and the current node itself |
| Following | Selects everything in the document after the closing  tag of the current node |
| following-sibling | Selects all siblings after the current node |
| Namespace | Selects all namespace nodes of the current node |
| Parent | Selects the parent of the current node |
| Preceding | Selects all nodes that appear before the current node  in the document, except ancestors, attribute nodes and namespace nodes |
| preceding-sibling | Selects all siblings before the current node |
| Self | Selects the current node |

**child vs descendant**

**child only deisplay child records not descendant will show all the stuff inside child as well**

**$x("(//div[contains(@id,'id1')]/descendant::\*)")**

**Page Facotry**

Selenium Page Factory Pattern is like an extension to [Page Object Model](http://www.seleniumeasy.com/selenium-tutorials/page-object-model-framework-introduction), but Page Factory is much enhanced model. To start with, we just need to import package ‘org.openqa.selenium.support.PageFactory’

"Factory class can be used to make using Page Objects simpler and easier".

We use [Page Factory pattern](https://selenium.googlecode.com/git/docs/api/java/org/openqa/selenium/support/PageFactory.html) to initialize web elements which are defined in Page Objects.

We should initialize page objects using initElements() method from PageFactory Class as below, Once we call initElements() method, all elements will get initialized. PageFactory.initElements() static method takes the driver instance of the given class and the class type, and returns a Page Object with its fields fully initialized.

Home homePage = **new** HomePage(driver);

PageFactory.initElements(driver, homePage);

Or,

*// To initialize elements.*

HomePage homePage = PageFactory.initElements(driver, HomePage.**class**);

Or, **as a constructor for page class as below:**

**public** **HompePage**(WebDriver driver) {

**this**.driver = driver;

PageFactory.initElements(driver, **this**);

}

@FindBy(name="username")

@CacheLookup

**private** WebElement userName;

What is @CacheLookup annotation in PageFactory?

We will mark ***annotation @CacheLookup*** to WebElements to indicate that it never changes (that is, that the same instance in the DOM will always be used)  
CacheLookup attribute can be used to instruct the InitElements() method to cache the element once its located and so that it will not be searched over and over again – this is useful when the elements that are always going to be there

**Difference between @Factory and @DataProvider**

Below is the main **difference between @Factory and @DataProvider** functionalities on TestNG.

1. **DataProvider**: A test method that uses DataProvider will be executed a multiple number of times based on the data provided by the DataProvider. The test method will be executed using the same instance of the test class to which the test method belongs.
2. **Factory**: A factory will execute all the test methods present inside a test class using a separate instance of the respective class.

**@DataProvider Example**

The below class contains the testMethod and beforeClass methods. testMethod takes a String argument and the value of the argument is provided by the DataProvider method, dataMethod. The beforeClassmethod prints a message onto the console when executed, and the same is the case with testMethod. testMethod prints the argument passed onto it to the console when executed.

|  |
| --- |
| public class DataProviderClass  {      @BeforeClass      public void beforeClass() {          System.out.println("Before class executed");      }        @Test(dataProvider = "dataMethod")      public void testMethod(String param) {          System.out.println("The parameter value is: " + param);      }        @DataProvider      public Object[][] dataMethod() {          return new Object[][] { { "one" }, { "two" } };      }  } |

Let’s run above test.

|  |
| --- |
| Before class executed  The parameter value is: one  The parameter value is: two  PASSED: testMethod("one")  PASSED: testMethod("two") |

As you can see from the preceding test results the class beforeClass is executed only one time irrespective of how many times the test method is executed.

**@Factory Example**

The below class contains the testMethod and beforeClass methods. The constructor of the test class takes a String argument value. Both beforeClass and testMethod print a message onto console.

|  |
| --- |
| public class SimpleTest  {      private String param = "";        public SimpleTest(String param) {          this.param = param;      }        @BeforeClass      public void beforeClass() {          System.out.println("Before SimpleTest class executed.");      }        @Test      public void testMethod() {          System.out.println("testMethod parameter value is: " + param);      }  }    public class SimpleTestFactory  {      @Factory      public Object[] factoryMethod() {          return new Object[] {                                  new SimpleTest("one"),new SimpleTest("two")                              };      }  } |

Let’s run above test.

|  |
| --- |
| Before SimpleTest class executed.  testMethod parameter value is: one  Before SimpleTest class executed.  testMethod parameter value is: two  PASSED: testMethod  PASSED: testMethod |

# Verify the Heght and Width

Generally i will use like below

driver.get("http://docs.seleniumhq.org/");

int width=driver.findElement(By.tagName("img")).getSize().getWidth();

int hight=driver.findElement(By.tagName("img")).getSize().getHeight();

System.out.println(width +">>>"+hight);

//to verify width

Assert.assertEquals(width, 200);

**Advantages of POM**

1. Page Object Patten says operations and flows in the UI should be separated from verification. This concept makes our code cleaner and easy to understand.
2. Second benefit is the **object repository is independent of testcases**,
3. Code becomes less and optimized because of the reusable page methods in the POM classes.

# Automation Framework Should following Qulaities:

1.Reusable-- Used by multiple projects and mutuple team

2.Scalable - othet framweowrk should also work within like tetsng,Jbheave,cucucmber and also for smal to large projects

3.Maintainable- maintenace should be easy not hard

4.Workable --anyone can use

Types of Framework

1. Data Driven

2. KeyWord driven

3.Hybrid

**Ques 3) How do you handle https website in selenium** **?By changing Firefox Profile**

public static void main(String[] args){

FirefoxProfile profile = new FirefoxProfile();

profile.setAcceptUntrustedCertificates(false);

WebDriver driver = new FirefoxDriver(profile);

driver.get("url");

 }

http://toolsqa.com/selenium-webdriver/event-listener/

**EventFiringWebDriver** is a class and is used to wrap your webdriver around to throw events and WebDriverEventListner is an interface which you have to implement to catch the webdriver events.Oc

You must have always craved for more logs from the Webdriver so that you can debug your scripts or may be log more information about your tests. Here is your answer to it, ***EventFiringWebDriver***and the***WebDriverEventListner*.***EventFiringWebDriver*is a class and is used to wrap your *webdriver*around to throw events and *WebDriverEventListner* is an *interface* which you have to implement to catch the *webdriver* events. Don’t worry if you didn’t understand what I just wrote, I will explain you in steps.

First thing to understand is your *EventFiringWebDriver* class. This is a class that implements the WebDriver interface. What the hell does that mean? It means that you will get all your regular webdriver methods like

*– findElementById*

*– findElementByTagName*

In addition to this you will have two methods

*– register(WebDriverEventListener eventListener)*

*– unregister(WebDriverEventListener eventListener)*

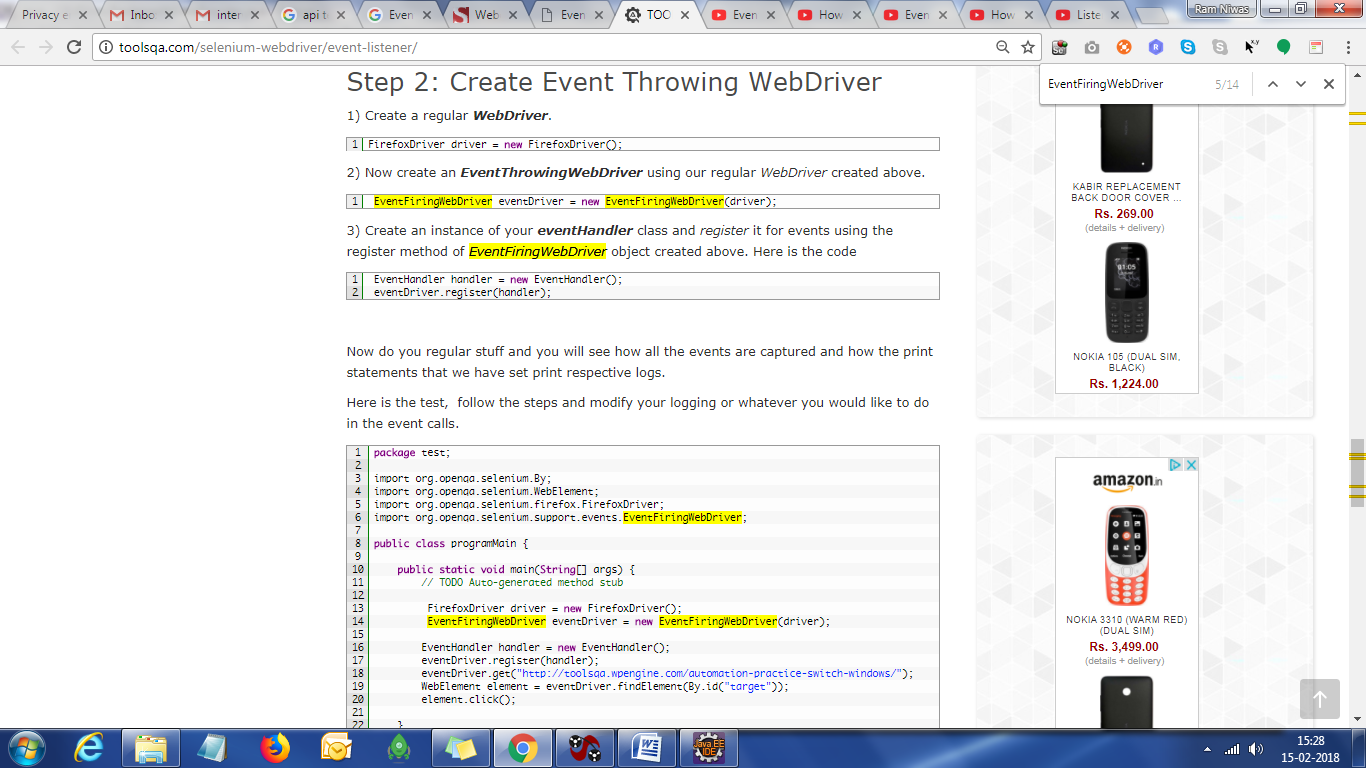
***Register***method will let you *register* our implementation of *WebDriverEventListner*to listen to the WebDriver *events* and with ***unregister***you will be able to *detach*.

Now firstly we have to initialize 'EventFiringWebDriver' by passing webdriver instance and then we need to register event listener for EventFiringWebDriver instance.

We also need to provide implementation for the events. We can do this by two ways, which ever is needed for our project.

##### **One way is by using**[**WebDriverEventListener Interface**](https://selenium.googlecode.com/git/docs/api/java/org/openqa/selenium/support/events/WebDriverEventListener.html)**If we use this, it will implement all the events that are supported by Webdriver.**

An example for this is shown below, when we use this, it will implement all the events like 'beforeNavigateTo', 'afterNavigateTo', 'beforeClickOn' , 'afterClickOn', 'onException' and more such events



**Ques 38) What are the different attributes for @Test annotation ?**

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

|  |
| --- |
| public class ExceptionTestDemo  {      @Test(expectedExceptions = { IOException.class }, expectedExceptionsMessageRegExp = "Pass Message test")      public void exceptionTestOne() throws Exception {          throw new IOException("Pass Message test");      }        @Test(expectedExceptions = { IOException.class }, expectedExceptionsMessageRegExp = ".\* Message .\*")      public void exceptionTestTwo() throws Exception {          throw new IOException("Pass Message test");      }        @Test(expectedExceptions = { IOException.class }, expectedExceptionsMessageRegExp = "Pass Message test")      public void exceptionTestThree() throws Exception {          throw new IOException("Fail Message test");      }  } |

Output of above test run is given below:

|  |  |
| --- | --- |
| [TestNG] Running: C:\Users\somepath\testng-customsuite.xml    PASSED: exceptionTestOne  PASSED: exceptionTestTwo  FAILED: exceptionTestThree  **We can use these annotations in those cases when we have more than a single criteria to to identify one or more WebElement objects.**  **@FindBys :** When the required WebElement objects need to match all of the given criteria use @FindBys annotation  **@FindAll :** When required WebElement objects need to match at least one of the given criteria use @FindAll annotation  Usage:  // FindBys annotations works as AND operator. It will return a list of all web elements if if satisfies all findBy condition.  @FindBys  ({  @FindBy(id = "ABC"),  @FindBy(name = "DEF"),  @FindBy(xpath = "XYZ")  })  private List<WebElement> MultipleInputElements;  Here List elementsWithBothclass1ANDclass2 will contain any WebElement which satisfies both criteria.  @FindAll({  @FindBy(className = "class1")  @FindBy(className = "class2")  })  private List<WebElement> elementsWithEither\_class1ORclass2  Here List elementsWithEither\_class1ORclass2 will contain all those WebElement that satisfies any one of the criteria.  Now what **cachelookup** does is it stores elements having @cachelookup annotation applied over it and then stores this element for further reference/s. **For example**:  public class SearchPage {  // The element is now looked up using the name attribute,  // and we never look it up once it has been used the first time  @FindBy(how = How.NAME, using = "q")  e @CacheLookup  private WebElement searchBox;  public void searchFor(String text) {  // We continue using the element just as before  searchBox.sendKeys(text);  searchBox.submit();  } }  **85) Using Selenium how can you handle network latency ?**  To handle network latency you can use driver.manage.pageloadingtime for network latency  **Customize Locator**  Using Java  import java.util.List;  import org.openqa.selenium.By;  import org.openqa.selenium.SearchContext;  import org.openqa.selenium.WebElement;  public class ByImageSrc extends By  {  private final String imageByString;  public ByImageSrc(String imageByString)  {  this.imageByString = imageByString;  }  @Override  public List<WebElement> findElements(SearchContext context)  {  List<WebElement> mockElements = context.findElements(By.xpath("//img[@src='" + imageByString + "']"));  return mockElements;  }  }  **//////////////////////////////**  usage :  **WebElement element = driver.findElement(new ByImageSrc("/path/to/image"));**   |  | | --- | |  | |

# Starting Test Automation for a Legacy Project

The questions vary from where to start, how much to automate and deciding on the best strategy for automation

" You can never test everything so you have to choose where you want to spend your time and money. Testing for me is primarily about information and risk and not about blanket coverage or completeness."

 Some of his suggested questions include,

* Where is the client currently experiencing the poorest quality?
* What are the key areas of functionality (e.g. what makes them money)?
* Would the client prefer more features to better quality?
* What are the biggest risks to their systems?
* If the client could improve one thing what would it be?
* Would exploratory manual testing find more important bugs?

**Selenium Query Commands for cookies**

**In Selenium Webdriver, we can query and interact with cookies with below built-in method:**

driver.manage().getCookies(); // Return The List of all Cookies

driver.manage().getCookieNamed(arg0); //Return specific cookie according to name

driver.manage().addCookie(arg0); //Create and add the cookie

driver.manage().deleteCookie(arg0); // Delete specific cookie

driver.manage().deleteCookieNamed(arg0); // Delete specific cookie according Name

driver.manage().deleteAllCookies(); // Delete all cookies

# [Run Parallel and Sequential tests](https://stackoverflow.com/questions/27694222/run-parallel-and-sequential-tests) both few classes in Parallel and few in sequence

<suite name="Suite1" verbose="1" >

<test name="ParallelGroup" parallel="methods" >

<classes>

<class name="testngtests.TestParallel"/>

</classes>

</test>

<test name="Sequential" parallel="false" >

<classes>

<class name="testngtests.copy.TestSeq"></class>

</classes>

</test>

</suite>

Set<Cookie> cookies = driver.manage().getCookies();

System.out.println("Size: " + cookies.size());

Iterator<Cookie> itr = cookies.iterator();

while (itr.hasNext()) {

Cookie cookie = itr.next();

System.out.println(cookie.getName() + "\n" + cookie.getPath()

+ "\n" + cookie.getDomain() + "\n" + cookie.getValue()

+ "\n" + cookie.getExpiry());

}

} catch (Exception e) {

e.printStackTrace();

**JavaScriptexecutor**

http://www.ufthelp.com/2014/11/what-is-javascriptexecutor-in-selenium.html

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript(Script,Arguments);

js.executeScript("window.location = 'http://demo.guru99.com/'");

WebElement button =driver.findElement(By.name("btnLogin"));

js.executeScript("arguments[0].click();", button);

CLick on hidden button

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("arguments[0].click();", element);

Scenario 4: Horizontal scroll on the web page.

**WebElement Element = driver.findElement(By.linkText("VBScript"));**

**//This will scroll the page Horizontally till the element is found**

**js.executeScript("arguments[0].scrollIntoView();", Element);**

Scenario 3: To scroll down the web page at the bottom of the page.

**js.executeScript("window.scrollTo(0, document.body.scrollHeight)");**

# [What is the use of “arguments[0]” when implementing javascriptexecutor?](https://stackoverflow.com/questions/40669291/what-is-the-use-of-arguments0-when-implementing-javascriptexecutor)

Just consider it as a command line arguments for your script. ; It is a reference to the arguments you pass in. In this case the index is 0 because you're passing in the element reference as the 0th argument in the executeScript call (the parameter after the String containing the script).

WebElement ele1=edriver.findElement(By.*id*("lst-ib"));

ele1.sendKeys("tdajdlajdljaest");

WebElement ele2=edriver.findElement(By.*name*("btnK"));

JavascriptExecutor js = (JavascriptExecutor)*driver*;

js.executeScript("arguments[1].click();",ele1,ele2); means it arguments[1] will take second parmeter and arguments[0] first

**Remote vs webdriver**

**https://community.perfectomobile.com/posts/992642-remotewebdriver-vs-webdriver**

The 'data:,' URL is just the default address that chromedriver navigates to when launching chrome. So this by itself doesn't necessarily mean that anything is going wrong.

* RemoteWebDriver is an implementation class of the WebDriver interface that a test script developer can use to execute their test scripts via the RemoteWebDriver server on a remote machine.
* There are two parts to RemoteWebDriver: a server and a client
* The RemoteWebDriver server is a component that listens on a port for various requests from a RemoteWebDriver client. Once it receives the requests, it forwards them to any of the following: Firefox Driver, IE Driver, or Chrome Driver, whichever is asked.

[**WebDriver**](http://www.gcreddy.com/2014/07/selenium-webdriver.html) is actually interface and implementations are firefoxdriver, iedriver, chromedriver, htmlunitdriver, remoteWebDriver

If tests are running on local browser then you can use all except RemoteWebDriver. If tests are running on remote machine’s browser then use

Capabilities & ChromeOptions

Capabilities are options that you can use to customize and configure a ChromeDriver session. This page documents all ChromeDriver supported capabilities and how to use them.

There are two ways to specify capabilities.

1. The first is to use the ChromeOptions class.
2. If your client library does not have a ChromeOptions class (like the selenium ruby client), you can specify the capabilities directly as part of the DesiredCapabilities.

**5.13 - Running TestNG programmatically**

You can invoke TestNG from your own programs very easily:

TestNG testng = new TestNG();

testng.setTestClasses(new Class[] { Run2.class });

TestListenerAdapter tla = new TestListenerAdapter();

testng.addListener(tla);

testng.run();

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| For example, suppose you want to create the following virtual file:   |  | | --- | | <suite name="TmpSuite" > | | <test name="TmpTest" > |  |  | | --- | | <classes> | | <class name="test.failures.Child"  /> | |  |  |  | | --- | --- | | <classes> | | | </test> |  |  | | --- | | </suite> |   You would use the following code:   |  |  | | --- | --- | | XmlSuite suite = new XmlSuite(); | | | suite.setName("TmpSuite"); |  |  | | --- | |  | | XmlTest test = new XmlTest(suite); | |  |  | | --- | | test.setName("TmpTest"); | | List<XmlClass> classes = new ArrayList<XmlClass>(); | |  |  |  | | --- | --- | | classes.add(new XmlClass("test.failures.Child")); | | | test.setXmlClasses(classes) ; |   And then you can pass this XmlSuite to TestNG:   |  |  | | --- | --- | | List<XmlSuite> suites = new ArrayList<XmlSuite>(); | | | suites.add(suite); |  |  |  | | --- | --- | | TestNG tng = new TestNG(); | | | tng.setXmlSuites(suites; |  |  | | --- | | tng.run(); | |

[**http://www.coffeecup.com/help/articles/absolute-vs-relative-pathslinks/**](http://www.coffeecup.com/help/articles/absolute-vs-relative-pathslinks/)

Absolute path contains full name of file including the source, for instance,

c:\Windows\Temp\log.txt. Relative path contains only the path relativaly to a

Relative path contains only the path relativaly to a

certain folder, for instance, relative path of log.txt relativaly to the folder Windows is Temp\log.txt.

**What is an absolute path?**

An absolute path is defined as the specifying the location of a file or directory from the root directory(/). In other words we can say absolute path is a complete path from start of actual filesystem from / directory.

Some examples of absolute path:

/var/ftp/pub

/etc/samba.smb.conf

/boot/grub/grub.conf

What is the relative path?

Relative path is defined as path related to the present working directory(pwd). Suppose I am located in /var/log and I want to change directory to /var/log/kernel. I can use relative path concept to change directory to kernel

changing directory to /var/log/kernel by using relative path concept.

pwd

/var/log

cd kernel

**Relative Paths**

* index.html
* /graphics/image.png
* /help/articles/how-do-i-set-up-a-webpage.html

**Absolute Paths**

* http://www.mysite.com
* http://www.mysite.com/graphics/image.png
* http://www.mysite.com/help/articles/how-do-i-set-up-a-webpage.html
* The first difference you'll notice between the two different types of links is that absolute paths *always* include the domain name of the website, including **http://www.**, whereas relative links only point to a file or a file path. When a user clicks a relative link, the browser takes them to that location on the current site. For that reason, you can only use relative links when linking to pages or files within your site, and you must use absolute links if you're linking to a location on another website.
* So, when a user clicks a relative link, how does their browser know where to take them? Well, it looks for the location of the file *relative* to the page where the link appears. (That's where the name comes from!) Let's get back to our first example:

|  |  |
| --- | --- |
| **Abbreviation** | **Expansion** |
| . | self::node() |
| .. | parent::node() |
| @ | attribute:: |
| \* | Selects all nodes of principal node type |
| // | /descendant-or-self::node()/ |

https://blogs.oracle.com/rajeshthekkadath/reverse-xpath-finding-the-xpath-of-a-element-using-bottom-up-approach

http://www.geekyarticles.com/2011/07/traversing-xpath.html

Reverse Travese in xpath

|  |  |
| --- | --- |
| **Sample XML Code** | **XPath** |
| https://cdn.app.compendium.com/uploads/user/e7c690e8-6ff9-102a-ac6d-e4aebca50425/f4a5b21d-66fa-4885-92bf-c4e81c06d916/Image/1882709d15c5cf640540383e24148570/xpath_reverse.png | (a) Using Top Down approach starting from a top node  ***//table/tr/a[text()='Login']***  (b) Directly accessing the child node using its text value ***//a[text()='Login']***  (c) Directly accessing the child node using child's attribute value ***//a[@href='1.html']***  (d) Directly accessing the child node using wildcard search. All occurences of text "Logi\\*" is matched by below xpath.It will work in this example since there is only one text "Login" which has occurence "Logi" in it.  ***//a[contains(text(),'Logi')]***  (e) Using Bottom Up approach starting from another child node and navigating to the actual node. Note: Here the DOM structure should be maintained (as it is in the xml) between the start node and end node  ***//a[text()='Home']/../../../table/tr/a[text()='Login']*** |

**Generic Selenium Questions**

**Question 6:**

Which language is used in Selenium IDE?

**Answer:**

Selenium IDE uses html sort of language called Selenese. Though other languages (java, c#, php ,Ruby ,Pearl, Python etc) cannot be used with Selenium IDE, Selenium IDE lets you convert test in these languages so that they could be used with Selenium 1.0 or Selenium 2.0

**Question 9:**

What are the element locators available with Selenium which could be used to locate elements on web page?

**Answer:**

There are mainly 8 locators used with Selenium –

**1         html id**

<div id="coolestWidgetEvah">...</div>

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

**2     html name**

<input name="cheese" type="text"/>

WebElement cheese = driver.findElement(By.name("cheese"));

**3       XPath locator and**

**4        Css locators**

<div id="food"><span class="dairy">milk</span><span class="dairy aged">cheese</span></div>

WebElement cheese = driver.findElement(By.cssSelector("#food span.dairy.aged"));

**5 Class Name**

<div class="cheese"><span>Cheddar</span></div><div class="cheese"><span>Gouda</span></div>

List<WebElement> cheeses = driver.findElements(By.className("cheese"));

**6. Tag Name**

<iframe src="..."></iframe>

WebElement frame = driver.findElement(By.tagName("iframe"));

**7. Link Text**

<a href="http://www.google.com/search?q=cheese">cheese</a>>

WebElement cheese = driver.findElement(By.linkText("cheese"));

**8. PartialLnkText**

<a href="http://www.google.com/search?q=cheese">search for cheese</a>>

WebElement cheese = driver.findElement(By.partialLinkText("cheese"));

**9 Using java script-**

You can execute arbitrary javascript to find an element and as long as you return a DOM Element, it will be automatically converted to a WebElement object.

WebElement element = (WebElement) ((JavascriptExecutor)driver).executeScript("return $('.cheese')[0]");

List<WebElement> labels = driver.findElements(By.tagName("label"));

List<WebElement> inputs = (List<WebElement>) ((JavascriptExecutor)driver).executeScript(

"var labels = arguments[0], inputs = []; for (var i=0; i < labels.length; i++){" +

"inputs.push(document.getElementById(labels[i].getAttribute('for'))); } return inputs;", labels);

**Question 10:**

What is Selenium Grid?

**Answer:**

Selenium grid lets you distribute your tests on multiple machines and all of them at the same time. Hence you can execute test on IE on Windows and Safari on Mac machine using the same test script (well, almost always). This greatly helps in reducing the time of test execution and provides quick feedback to stack holders.

**Question 24:**

What is CSS location strategy in Selenium?

**Answer:**

CSS location strategy can be used with Selenium to locate elements, it works using cascade style sheet location methods in which -

Direct child is denoted with – (a space)

Relative child is denoted with - >

Id, class, names can also be used with XPath –

         css=input[name=’q’]

         css=input[id=’lst-ib’] or input#lst-ib

         css=input[class=’ lst’] or input.lst

If only part of id/name/class is constant than “contains” can be used as –

         css=input[id\*=' lst-ib ')]

Element location strategy using inner text

         css = a:contains(‘log out’)

Question 29:

Can I handle java script alert using Selenium?

Answer

You could use verify/assertAlert to check presence of alert on page. Since selenium cannot click on “Ok” button on js alert window, the alert itself does not appear on page when this check is carried out.

**Question 42:**

How do I verify presence of drop down options using Selenium?

**Answer**

Use assertSelectOptions as following to check options in a drop down list –

assertSelectOptions

// Get a handle to the open alert, prompt or confirmation

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

// Get the text of the alert or prompt

alert.getText();

// And acknowledge the alert (equivalent to clicking "OK")

alert.accept();

**alert.dismiss()**

//will get the text which is present on th Alert.  
**alert.getText();**

//Will pass the text to the prompt popup  
**alert.sendkeys();**

**Selenium RC (Selenium 1.0) Questions**

**What is the difference between captureEntirePageScreenshot and CaptureScreenShot?**

Here discussed about difference between **CaptureEntirePageScreenshot** and CaptureScreenShot. CaptureEntirePageScreenshot captures the AUT web page only. CaptureScreenShot captures the System screen shot.

include test in test xml

<test verbose="2" name="com.src.testng.MercTestNgSuite" junit="false" annotations="JDK">

<classes>

<class name="com.src.testng.MercTestNgSuite">

<methods>

<include name="testLogin1"/>  
<include name="testFindFlights"/>  
<include name="testSelectFlights"/>  
<include name="testFillUserDetails"/>  
<include name="testVerifyFlightConf"/>  
<exclude name="testLogout"/>

</methods>

</class>

</classes>

</test>

**Question 53:**

Why do I need Selenium Server?

**Answer**

Selenium uses java script to drives tests on a browser; Selenium injects its own js to the response which is returned from aut. But there is a java script security restriction (***same origin policy***) which lets you modify html of page using js only if js also originates from the same domain as html. This security restriction is of utmost important but spoils the working of Selenium. This is where Selenium server comes to play an important role.

Selenium server stands between aut and browser and injects selenium js to the response received from aut and then it is delivered to broswer. Hence browser believes that entire response was delivered from aut.

Selenium RC injects javascript function into browsers when the web page is loaded.

Selenium WebDriver drives the browser using browser's built-in

**Question 54:**

What is Selenium core? I have never used it!!!

**Answer**

Selenium core is the core js engine of Selenium which executes tests on browser, but because of same origin policy it needs to be deployed on app server itself, which is not always feasible. Hence Selenium core is not used in isolation. Selenium IDE as well as Selenium RC use Selenium core to drive tests while over coming same origin policy. In case of Selenium IDE tests are run in context of browser hence it is not hindered by same origin policy and with Selenium RC, Selenium Server over comes same origin policy.

**Question 61:**

I am not using java to program my tests, do I still have to install java on my system?

**Answer**

Yes, since Selenium server is written in java you need java to be installed on your system to be able to use it. Even though you might not be using java to program your tests.

**Question 73:**

How do I iterate through options in my test script.

**Answer**

You can use loop features of the programming language, for example you can use “for” loop in java as following to type different test data in a text box –

             // test data collection in an array

            String[] testData = {"test1", "test2", "test3"};

     // iterate through each test data

**for** (String s : testData) {

              selenium.type(“elementLocator”, testData);

     }

String[] exp = {"--Title--","Mr","Mrs","Miss","Ms","Dr","Prof"};

WebElement dropdown = driver.findElement(By.id("ddlNights"));

Select select = new Select(dropdown);

**List<WebElement> options = select.getOptions();**

for(WebElement we:options)

{

boolean match = false;

for (int i=0; i<exp.length(); i++){

if (we.getText().equals(exp[i]){

match = true;

}

}

Assert.assertTrue(match);

}

I

**Question 74:**

Can I execute java script from my tests?

// I want to count number of images on my page.

**Answer**

You can use method getEval() to evaluate java script. For example if you want to count number of images then you can pass following dom statement to getEval() as following –

            selenium.getEval("window.document.images.length;");

Or to get All anchor objects from a page

            selenium.getEval("window.document.getElementsByTagName(‘a’);");

**Question 79:**

My application has lots of pop up window, how do I work with them?

**Answer**

You need to know the Window ID of pop window to be able to work with them.

First you need to bring control on pop up window; execute selenium commands there, close the pop up window and then bring control back to main window. Consider following example where click on an image brings a pop up window –

                         // click on image brings pop up window

selenium.click("css=img");

// wait for pop up window identified using anchor target "ss"

          selenium.waitForPopUp("ss", getWaitPeriod());

          selenium.selectWindow("ss");

          // Some more operations on popup window

// Close the pop up window and Select the main application window

          // Main window is selected by adding null as argument

          selenium.close();

          selenium.selectWindow("null");

          // continue with usual operation 

Question 81:

My application uses Ajax heavily how do I use Selenium RC to work with Ajax operations?

**Answer**

Ajax operations don’t reload a page like normal form submission but they make http requests behind the scene. You cannot use waitForPageToLoad for such operations and instead should use conditional wait for change in state of application. This could as well mean waiting for presence of an element before continuing with test operations. Consider following example in which type operation triggers Ajax operation which is followed by conditional wait for presence of a text box –

// type operation brings element “q” on screen without loading the page

selenium.type("elementLocator", "testData");

// conditional wait for element “q”

**for** (**int** second = 0;; second++) {

**if** (second >= 60) *fail*("timeout");

**try** { **if** (selenium.isElementPresent("q")) **break**; } **catch** (Exception e) {}

          Thread.*sleep*(1000);

}

**Question 82:**

How do I upload a file using Selenium? I need to upload a word file during test execution.

**Answer**

If you are using Firefox then you can use “type” command to type in a File Input box of upload file. But type operation does not work with IE and you would have to use “Robot” class in java to work make file upload work.

# How To Get X Y Coordinates Of Element In Selenium WebDriver

WebElement Image = getDriver().findElement(By.xpath("//img[@border='0']"));

Point point = Image.getLocation();

**int** xcord = point.getX(); //suppose 3

System.***out***.println("Element's Position from left side Is "+xcord +" pixels.");

**int** ycord = point.getY();//suppose 6

System.***out***.println("Element's Position from top side Is "+ycord +" pixels.");

Actions action = **new** Actions(getDriver());

action.moveToElement(Image, xcord, ycord).click().build().perform();

**Robot class**

Robot robot = new Robot(); // Robot class throws AWT Exception

Thread.sleep(2000); // Thread.sleep throws InterruptedException

robot.keyPress(KeyEvent.VK\_DOWN);

**Question 92:**

How does Selenium grid works?

**Answer**

Selenium grid uses combination of Selenium RC servers to execute tests in multiple browsers on different machine. Herein one Selenium RC server works as hub while other RC servers work as slaves, which could be controlled by hub. Whenever there is a request for a specific configuration for test execution then hub looks for a free RC slave server and if available then test execution begins on it. Once test execution is over then RC slave server would be available for next set of test execution.

**Question 94:**

Which jar files are needed to works with Selenium GRID?

**Answer**

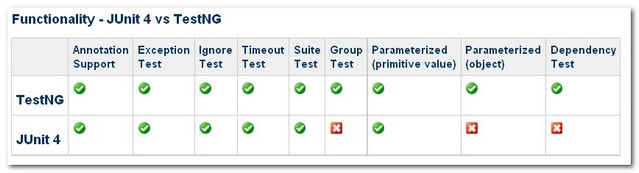
You need to download and add following jar files to your Selenium set up to be able to work with Selenium. These jar files are –

selenium-grid-remote-control-standalone-.jar

         selenium-grid-hub-standalone-.jar

         selenium-grid-tools-standalone-.jar

|  |  |  |
| --- | --- | --- |
| **Selenium IDE** | **Selenium RC** | **Selenium WebDriver** |
| **It only works in Mozilla browser.** | **It supports with all browsers like Firefox, IE, Chrome, Safari, Opera etc.** | **It supports with all browsers like Firefox, IE, Chrome, Safari, Opera etc.** |
|  |  |  |
| **Doesn’t required to start server before executing the test script.** | **Required to start server before executing the test script.** | **Doesn’t requireto start server before executing the test script.** |
| **It is a GUI Plug-in** | **It is standalone java program which allow you to run Html test suites.** | **It actual core API which has binding in a range of languages.** |
| **Core engine is JavaScript based** | **Core engine is JavaScript based** | **Web Driver is faster than Selenium RC since it speaks directly to the browser uses the browser’s own engine to control it.** |
| Very simple to use as it is record & playback. | It is easy and small API | As compared to RC, it is bit complex and large API. |
| **It is not object oriented** | **API’s are less Object oriented** | **API’s are entirely Object oriented** |
| **It doesn’t supports of moving mouse cursors.s** | **It doesn’t supports of moving mouse cursors.** | **It supports of moving mouse cursors.** |
| **It does not supports listeners** | **It does not supports listeners** | **It supports the implementation of listeners** |
| It does not support to test iphone/Android applications | It does not support to test iphone/Android applications | It support to test iphone/Android applications |



# *Listeners*

Listener is defined as interface that modifes the default TestNG's behavior. As the name suggests Listeners "listen" to the event defined in the selenium script and behave accordingly. It is used in selenium by implementing Listeners Interface.

|  |
| --- |
| IAnnotationTransformer -  This method will be invoked by TestNG to give you a chance |
| to modify a TestNG annotation read from your test classes. |

|  |  |
| --- | --- |
| You can change the values you need by calling any of the | |
| setters on the ITest interface.  1. IAnnotationTransformer  **only lets you modify a @Test annotation. If you need to modify another TestNG annotation (a configuration annotation, @Factory or @DataProvider) use IAnnotationTransformer2.** |
|  |
|  |

2. IAnnotationTransformer2 -  This method will be invoked by TestNG to give you a chance to modify a

TestNG annotation read from your test classes.

1. IConfigurable ,
2. IConfigurationListener ,
3. IExecutionListener,
4. IHookable ,
5. IInvokedMethodListener ,'

 The working of this listener is also exactly the same as ISuiteListerner & ITestListerner and the only difference is that it makes the call before and after every Method. It has only two methods in it.

**afterInvocattion():**Invoke after each method

**beforeInvocation():** Invoke before each method

1. IInvokedMethodListener2 ,
2. IMethodInterceptor ,
3. **IReporter**, generateReport
4. **ISuiteListener**,It has two method in it **onStart()** & **onFinish()**. Whenever a class implements this listener, TestNG guarantees the end-user that it will invoke the methods onStart() and onFinish() before and after running a TestNG Suite. So before TestNG picks up your suite for execution, it first makes a call to onStart() method and runs whatever has been scripted in this method. In a similar way, it again makes a call to onFinish() method after a suite has been run.
5. **ITestListener** for screenshot OnTestFail, on testPass, onTestSKip
6. **IRetryAnalyzer** for retry methods set count to retry limit

A *listener* is called when the user does something to the user interface that causes an event

Listeners gives us the ability to act before and after of every Suite, Test and Methods.

To say it in simple words, we can make TestNG to listen what we say with the help of Listeners. Listeners give us the flexibility to modify default TestNG's behaviors.

By using TestNG listeners 'ITestListener' or 'TestListenerAdapter' we can change the default behaviour write our own implementation when a Test fails or Skips etc.

what does they do?   
listeners gives the flexibility for the user to listen to the events that are generated from the start of the execution of script till the end.  
  
what kind of events?  
events such as, when a test method goes into one of passed/failed/skipped state.  
events such as, when a configuratiom annotated method goes into one of passed/failed/skipped state.  
events such as before and after a annotated method is executed.  
  
what is the actual purpose?   
the main intention for providing listeners is to extend the default reports behaviour of TestNg. Extend this feature to generate your own custom reports by listening to the above events.  
  
what else can you do?  
unlimited possibilities: Few imp things such as,  
**Before the start of execution, Check if the server is up and running  
Sending an automatic status mail at the end of execution  
Any other task of your need**

# Why Selenium Server not required by Selenium WebDriver?

http://www.softwaretestingclass.com/why-selenium-server-not-required-by-selenium-webdriver/

In previous article we have seen about [Difference between selenium IDE, RC & WebDriver](http://www.softwaretestingclass.com/difference-between-selenium-ide-rc-webdriver/).  
As we already know Selenium RC requires starting the server to kick of the execution of **Selenium Automated RC** test suites. The Selenium server is the intermediator between browser and Selenium RC because Selenium RC won’t make direct call to browser. So we have to start Selenium server prior to start running Selenium RC test cases.  
In case of **Selenium WebDriver**, it does not required to start Selenium Server for executing test scripts. Selenium WebDriver makes the calls between browser & automation script. Selenium WebDriver has native supports for each browser to supports Test Automation; on same machine (**both Selenium WebDriver Automation tests & browsers are on same machine.**)

But it might be possible to use Selenium server while running Selenium WebDriver test scripts if both Selenium WebDriver Automation tests & browsers are on different machine. This is possible while using Selenium Grid, because of executing different test scripts simultaneously on different machines to save the execution time. If we have to execute the test suite having 5 independent test cases that might take 5 days to finish, so we use Selenium Grid with 5 machines & execute test cases simultaneously to finish the automation test cases in 1 day.

**Question 116:**

How is Selenium 2.0 configuration different than Selenium 1.0?

**Answer**

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.

**Question 127:**

What is the order of fastest browser implementation for WebDriver?

**Answer**

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.

**Question 131:**

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

**Answer**

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.

**Question 133:**

Is there any difference in XPath implementation in different WebDriver implementations?

**Answer**

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.

**Question 140:**

Is it possible to interact with hidden elements using WebDriver?

**Answer**

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.

Webelement element=driver.findele……..

Indicates that a driver can execute JavaScript, providing access to the mechanism to do so.

**To click on any hidden element**

String Script = "javascript:document.getElementById('testobject').click();";

((JavascriptExecutor) webdriver).executeScript(**Script**);

WebElement invisibleelement= driver.findElement(ElementLocator);

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("arguments[0].click();", invisibleelement);

**To getText on any hidden element**

String text=js.executeScript("return arguments[0].innerHTML", element);

***Clicking an invisible or hidden Element***

List<WebElement> number\_of\_display\_none = driver.findElements(By.cssSelector("div[style\*='none']"));

JavascriptExecutor executor = JavascriptExecutor)driver;

executor.executeScript("arguments[0].click();", number\_of\_display\_none.get(position).getText());

**OR**

WebElement invisibleelement= driver.findElement(ElementLocator);

JavascriptExecutor executor = (JavascriptExecutor)driver;

executor.executeScript("arguments[0].click();", invisibleelement);

[Email This](https://www.blogger.com/share-post.g?blogID=9218489852882647802&postID=8740911290437650925&target=email)

//If id is given then can click like this on hidden element

String Script = "javascript:document.getElementById('loginbutton').click();";

((JavascriptExecutor) *driver*).executeScript(Script);

**Question 142:**

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

**Answer**

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

WebDriver does not allow for interacting with hidden elements. It makes sense - user cannot click a hidden field using Sizzle Library and not all the capabilities like “contains” are available to be used with Selenium 2.0

**Question 145:**

Are there any limitations from operating systems while using WebDriver?

**Answer**

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

**Question 151:**

What are the modes of Remote WebDriver

**Answer**

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.

**Question 154:**

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

**Answer**

Use this –

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

     driver.setJavascriptEnabled(**true**);

or this –

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

**Question 155:**

Is it possible to emulate a browser with HTMLUnitDriver?

**Answer**

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);

**Question 158:**

Can WebDriver handle UntrustedSSLCertificates?

DesiredCapabilities capability = DesiredCapabilities.chrome(); *// To Accept SSL certificate* capability.setCapability(CapabilityType.ACCEPT\_SSL\_CERTS, **true**); *// setting system property for Chrome browser*

System.setProperty("webdriver.chrome.driver", "E:/chromedriver.exe");

**Question 160:**

How do I simulate keyboard keys using WebDriver?

**Answer**

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

**Question 161:**

What about Mouse Interaction?

**Answer**

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.

**Question 165:**

What is grid2?

**Answer**

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

**Selenium Tool Implementation Misc Questions**

**Question 169:**

How do I implement data driven testing using Selenium?

**Answer**

Selenium, unlike others commercial tools does not have any direct support for data driven testing. Your programming language would help you achieving this. You can you jxl library in case of java to read and write data from excel file. You can also use Data Driven Capabilities of TestNG to do data driven testing.

**Question 170:**

What is equivalent to test step, test scenario and test suite in Selenium.

**Why CSS locator is better than Xptah**

* Xpath engines are different in each browser, hence make them inconsistent
* IE does not have a native xpath engine, therefore selenium injects its own xpath engine for compatibility of its API. Hence we lose the advantage of using native browser features that WebDriver inherently promotes.

 css 's an HTML standard.  While XPath on the other hand is an XML  standard, so

**Describe some problems that you had with Selenium tool ?**

1 Windows Popup

2 Testing Flash apps: To automate flash apps with Selenium, one can use Flex Monkium.

Solution: The one big limitation in Selenium - Select a file in a File Input field for sake of uploading.  
XPaths makes tests slow  
Having Ids for elements make tests faster but the UI code might not have Ids for all elements  
XPaths are brittle - ie flaky because if the UI developer changes anything like adding or removing a DIV, then the XPath becomes invalid and has to be updated in the tests.  
\*Also look at my blog above which talks more about File Upload issue with Selenium \*

## 11. How to run test cases with dependent in Selenium using TestNG?

Wed, 05/04/2011 - 23:29 — [Srinivasa Bittla](http://qualitytrainings.com/user/3)

The @Test should be followed by (dependsOnMethods = "testLogin")  
Note:- The test case will be executed after the testLogin case

Ex: @Test(dependsOnMethods = "testLogin")

## 18. What are the limitations of selenium RC

The limitations of selenium RC are:  
1) Switching between the multiple instances of the same browser is not possible  
2) Switching between the multiple instances of the different browsers is not possible  
3) Browser navigation, like back and forward button emulations is not possible  
4) Limited features in terms of drag and drop of objects  
5) To work with Ajax based UI elements there are very limited features are there with Selenium RC

To overcome the above limitations we use selenium web driver or google web driver

## 19. How to do database testing using selenium rc?

There are no selenium specific commands to do the database testing. But, you have the work around with native language. Here is the example how you can make database testing in selenium. The below mentioned commands will help you to begin with.

Before you use the below code **You should have created the DSN name for** **MS SQL Server DB or Oracle DB using ODBC drivers. Here the name that we have used for the database DSN name is QT\_Flight32. And the table in that database we are using is Orders. Hope the remaining steps you can easily understand.**

**//KEEP THE NAME OF THE PACKAGE**  
package com.src.test;

**//IMPORT** **THE SQL PACKAGE  
import java.sql.\* ;**

**//DECLARE THE CLASS**  
class JDBCODBCTestCase  
{

**//WRITE THE MAIN METHOD TO CONNECT TO THE DATABASE**   
public static void main( String args[] )  
{

try{

**// LOAD THE DATABASE DRIVER**  
Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" ) ;

**// ESTABLISH THE CONNECTION TO THE DATABASE**  
Connection conn = DriverManager.getConnection( "jdbc:odbc:QT\_Flight32" ) ;

**// GET A STATEMENT FOR THE CONNECTION**  
Statement stmt = conn.createStatement() ;

**// PREPARE THE SQL STATEMENT**  
String strSQL = "SELECT \* FROM Orders";

**// EXECUTE THE SQL QUERY AND STORE IN RESULTS SET**  
ResultSet rs = stmt.executeQuery( strSQL ) ;

**// LOOP THE RESULT TILL IT REACHED END**

**while**(rs.next()) {

int EmpId= rs.getInt("EmpId");

**String** EmpName= rs.getString("EmpName");

**String** EmpAddress=rs.getString(3);

Double EmpSal= rs.getDouble(4);

**String** EmpDept=rs.getString("EmpDept");

System.out.println(EmpId+"\t"+EmpName+"\t"+EmpAddress+"\t"+EmpSal+"\t"+EmpDept);

}

**// rs.getString(1) - FIRST COLUMN**  
**// rs.getString(2) - SECOND COLUMN ETC**  
while( rs.next() )  
System.out.println( rs.getString(1) + rs.getString(2)+ rs.getString(3)+ rs.getString(4)) ;

**// CLOSE THE RESULT, STATEMENT AND CONNECTION**  
rs.close() ;  
stmt.close() ;  
conn.close() ;

}  
**// HANDLE THE SQL EXCEPTION**  
catch( SQLException se ) {

System.out.println( "SQL Exception:" ) ;

**// PRINT TILL ALL THE ECEPTIONS ARE RAISED**  
while( se != null ) {

System.out.println( "State : " + se.getSQLState() ) ;  
System.out.println( "Message: " + se.getMessage() ) ;  
System.out.println( "Error : " + se.getErrorCode() ) ;

se = se.getNextException() ;

}

}  
**//CATCH THE CLASS EXCEPTION**  
catch( Exception e ) {

System.out.println( e ) ;

}  
}

}

Updating Records

**Updating records:**  
To update records in a table we use a method called executeUpdate(String str)

int result = st.executeUpdate("**update** Employee **set** EmpName='Robert' **where** EmpId=2");

System.out.println("**number** **of** records updated **is**" +result);

Get Coulumn count

**int** columnCount = resultSet.getMetaData().getColumnCount();

**Access Priperty File**

**Properties extends HashTable implements Map, so you can get all the keys as a Set<String>using keySet().**

**Any property value is a String. You can split the String given a delimiter.**

File file = **new** File("config.properties");

FileInputStream fileInput = **new** FileInputStream(file);

Properties prop = **new** Properties();

prop.load(fileInput);

System.***out***.println(prop.keySet());

Set<Object> st =prop.keySet();

**for**(Object set:st)

{

System.***out***.println(set+"--"+prop.get(set));

//System.out.println();

}

}

try {

//load a properties file from class path, inside static method

prop.load(App.class.getClassLoader().getResourceAsStream("config.properties"));

'

  System.out.println(pro.keySet());

//get the property value and print it out

System.out.println(prop.getProperty("database"));

System.out.println(prop.getProperty("dbuser"));

System.out.println(prop.getProperty("dbpassword"));

}

catch (IOException ex) {

ex.printStackTrace();

}

### How to iterate Properties Files in Java?

### 1.Properties props = System.getProperties();

### for (String key : props .stringPropertyNames())

### {

### System.out.println(key + " = " + props .getProperty(key));

### }

### 2. Properties props = System.getProperties();

### Enumeration e = props.propertyNames();

### while (e.hasMoreElements())

### {

### String key = (String) e.nextElement();

### System.out.println(key + " = " + props.getProperty(key));

### }

### 3. Properties props = System.getProperties();

### SortedMap sortedSystemProperties = new TreeMap(props);

### Set keySet = sortedSystemProperties.keySet();

### Iterator iterator = keySet.iterator();

### while (iterator.hasNext())

### {

### String key = (String) iterator.next();

### System.out.println(key + " = " + props .getProperty(key));

### }

1DB Verification

*//Test to verify Employee table has a record with employee name 'Jack'*

    @Test(priority = 2)

**public** **void** **tesVerifyListOfRecords**() {

**boolean** flag = **false**;

        List<String> listOfDBValues = **new** ArrayList<String>();

        String expEmployeeName = "Jack";

        String sqlQuery = "select EmpName from employee";

*//Getting list of employee names from employee table*

        listOfDBValues = DataBaseConnector.executeSQLQuery\_List("QA", sqlQuery);

**for** (String strName : listOfDBValues) {

**if** (strName.equalsIgnoreCase(expEmployeeName)) {

                flag = **true**; **break**;

            } }

        Assert.assertTrue(flag, "Retrieved values are not matching with Expected values");   }}

# [implicit wait vs Explicit wair vs Thread.sleep()](http://stackoverflow.com/questions/13975556/selenium-implicit-wait-vs-thread-sleep)

**Difference is  
1. Obvious - Implicit wait time is applied to all elements in your  
script but Explicit only for particular element  
2. In Explicit you can configure, how frequently (instead of 500  
millisecond) you want to check condition.  
3. In Explicit you can also configure to ignore other exceptions than  
"NoSuchElement" till timeout..**

Implicit wait - It's global setting applicable for all elements and if element appear before specified time than script will start executing otherwise script will throw NoSuchElementException. Best way to use in setup method. Only affect By.findelement().

Thread. Sleep () - It will sleep time for script, not good way to use in script as it's sleep without condition.**\*Never\* use thread.sleep, it's not an explicit wait, it's a forced wait.**

Explicit wait is the wait which you normallly insert anywhere in your code .it is like this Thread.sleep(1000) for 1sec.  
Implict waits are set only once at the start of the code which is used to wait for any command during the run. for example if an element is not yet visible then selenium will wait till the time which is set expires. it is like this :

An explicit wait is where you explicitly define the condition you are waiting for and your code explicitly waits for that condition to occur before continuing.    
  
Stealing an example from the Selenium docs:

WebDriverdriver=**new**FirefoxDriver();

driver.get(["http://somedomain/url\_that\_delays\_loading"](http://somedomain/url_that_delays_loading));

WebDriverWait **wait=new** WebDriverWait(driver,**10**)

WebElementmy DynamicElement=wait

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

**Conditions**

1. ExpectedCondition < WebElement > **elementToBeClickable**(By locator)
2. Expected
3. Condition < Boolean > **elementToBeSelected**(By locator)
4. ExpectedCondition < WebElement > **presenceOfElementLocated**(By locator)
5. ExpectedCondition < Boolean > **titleContains**(String title)
6. ExpectedCondition < Boolean > **titleIs**(String title)
7. ExpectedCondition < Boolean > **urlContains**(String fraction)
8. ExpectedCondition < Boolean > **urlToBe**(String url)
9. ExpectedCondition < WebElement > **visibilityOfElementLocated**(By locator)

**Question 125:**

How about handling frames using Selenium 2.0?

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

**Question 126:**

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

driver.navigate().forward();

driver.navigate().back();

driver.navigate().refresh();

[**What are the types of Assertions there in Selenium? | Selenium**](http://blosumsdotnet.blogspot.in/2012/05/what-are-types-of-assertions-there-in.html)

Assertion is nothing but a check or verification point.  
**Selenium Assertions can be used in 3 modes:**  
**1) assert -**When an “assert” fails, the test will be aborted. If you are executing test suite, the next state case will start  
**2) verify -**When a “verify” fails, the test will continue execution, logging the failure.  
**3) waitFor -**“waitFor” commands wait for some condition to become true (which can be useful for testing Ajax applications). They will succeed immediately if the condition is already true. However, they will fail and halt the test if the condition does not become true within the current timeout setting.

**There are three types of Selenese, those are:**  
**1. Actions -** used for performing the operations and interactions with the target elements  
**2. Assertions -** used as check points  
**3.Accessors -** used for storing the values in a variable.

**There are many challenges with Selenium.**  
1. Selenium Supports only web based applications  
2. It doesn’t support any non web based (Like Win 32, Java Applet, Java Swing, .Net Client Server etc) applications  
3. When you compare selenium with QTP, Silk Test, Test Partner and RFT, there are many challenges in terms of maintainability of the test cases  
4. Since Selenium is a freeware tool, there is no direct support if one is in trouble with the support of applications  
5. There is no object repository concept in Selenium, so maintainability of the objects is very high  
6. There are many challenges if one have to interact with Win 32 windows even when you are working with Web based applications  
7. Bitmap comparison is not supported by Selenium  
8. Any reporting related capabilities, you need to depend on third party tools  
9. You need to learn any one of the native language like (.Net, Java, Perl, Python, PHP, Ruby) to work efficiently with the scripting side of selenium.

**Question 175:**

**Why should I use Page Object?**

Answer

Page object is a design pattern which distinguishes the code carrying out operations on page and code which carries out tests (assertion/verification). While implementing page object you abstract functioning of a page or part of it in a dedicated “Classs” which is then used by test script to perform actions on page and reach a stage when actual test could be performed.

Advantage of using page object is the fact that if application lay out changes then you only need to modify the navigation part and test would function intact.

Check If An Element Exists

driver.findElements(By.id("element-id")).size()!=0

Refresh Page

**driver.navigate().refresh()**;

Wait For Element To Be Available

WebDriverWait wait = new WebDriverWait(driver, 30);

WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id("id1)

How To Check If An Element Is Visible With WebDriver

**http://www.fromdev.com/2013/09/webdriver-selenium-code-snippets.html**

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

if(element instanceof RenderedWebElement) {

System.out.println("Element visible");

} else {

System.out.println("Element Not visible");

**}**

# [Use Selenium to find all “hoverable” elements on a webpage](http://stackoverflow.com/questions/17077662/use-selenium-to-find-all-hoverable-elements-on-a-webpage)

WebDriver driver = newFirefoxDriver();

driver.navigate().to("http://yourdomain.com/page.html");

List<WebElement> mouseoverElements = driver.findElements(By.xpath("//\*[@onmouseover]"));

System.out.println("Number of elements with a mouseover attribute: " + mouseoverElements.size());

for (WebElement e : mouseoverElements) {

System.out.println(e.getTagName() + ": " + e.toString());

}

**Count of Elements TextBox Selenium**

**No of text Boxes/pages/Elements/Images in Page- According to tag we can count**

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

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

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

List<WebElement> e1 = driver.findElements(By.*tagName*("a"));

System.*out*.println(e1.size());

**for** (WebElement e : e1)

{

System.*out*.println(e.getText());

}

**TextBox**

1. WebElement element  = driver.findElement(By.id("element-id"));
2. //Send empty message to element for setting focus on it.
3. element.sendKeys("");

**Drop Down**

**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 –

WebElement dropDownListBox =

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

WebElement dropDown =driver.findElement(By.id("edit-ticket-no"));

Select sel =new Select(dropDown);

sel.selectByVisibleText("1 ticket");

**Get All Elements with particual Attribute**

List<WebElement> ls=edriver.findElements(By.*xpath*("//\*[@class='darla']"));

System.***out***.println(ls.size());

**Get All options Element in Select**

Select oSelect = new Select(driver.findElement(By.id("yy\_date\_8")));

List <WebElement> elementCount = oSelect.getOptions();

int iSize = elementCount.size();

for(int i =0; i<iSize ; i++){

String sValue = elementCount.get(i).getText();

System.out.println(sValue);

**Windows handling**

**1**public void OpenBrowser(){

driver=new FirefoxDriver();

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

driver.manage().timeouts().implicitlyWait(30,TimeUnit.SECONDS);

driver.get(http://www.hdfcbank.com/);

}

@Test

public void windowSwitching(){

WebElement Loginbutton=driver.findElement(By.id("loginsubmit"));

String Parentwin=driver.getWindowHandle();

Loginbutton.click();

Set<String> allwin=driver.getWindowHandles();

for(String childwin:allwin){

if(!childwin.equals(Parentwin)){

driver.switchTo().window(childwin);

}

}

if(driver.getTitle().equals("netbanking")){

OR // By GetURL if(driver.getURL ().equals("https://accounts.google.com/ServiceLogin?hl=en&passive=true&continue")){

WebElement Continuebutton=driver.findElement(By.xpath("//img[@alt='Continue']"));

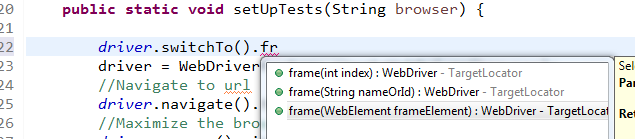
Continuebutton.click();

//Switch on frame

//driver.switchTo().frame(1); basis of index

//driver.switchTo().frame(1); basis of frame name

driver.switchTo().frame(driver.findElement(By.name("login\_page")));



WebElement button=driver.findElement(By.xpath("//img[@alt='continue']"));

button.click();

//Handling alert

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

System.out.println(alt.getText());

if(alt.getText().equals("Customer ID cannot be left blank.")){

alt.accept();

// alt.dismiss(); click on cancel button

driver.close();

driver.switchTo().window(Parentwin);

if(driver.getTitle().equals("HDFC Bank: Personal Banking Services")){

driver.quit();

}

}

else{ System.out.println("fail");}

}

else{ System.out.println("Switching fail");

}

**2 –Windows handling By Iterator**

**package** Core;

**import** java.util.Iterator;

**import**java.util.List;

**import** java.util.Set;

**import**java.util.concurrent.TimeUnit;

//import org.apache.jasper.tagplugins.jstl.core.Set;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import**org.openqa.selenium.support.ui.Select;

**publicclass** BrowserOpenClose {

**publicstaticvoid** main(String[] args) **throws** InterruptedException

{

/\* WebDriver driver= new FirefoxDriver();

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

WebElement sign = driver.findElement(By.id("//\*[@id='topBg']/div/div[1]/a[2]"));

sign.click();\*/

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

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

WebElement sign = driver.findElement(By.*id*("loginsubmit"));

sign.click();

Set<String> windowId = driver.getWindowHandles(); // get window id of current window

Iterator<String> itererator = windowId.iterator();

//make iterator for all winodows which has to close like we have now 3 windows and now want close 2 window

String mainWinID = itererator.next();

String newAdwinID = itererator.next();

// String newAdwinID1 = itererator.next();

driver.switchTo().window(newAdwinID);

System.*out*.println(driver.getTitle());

WebElement button=driver.findElement(By.*xpath*("//\*[@id='wrapper']/div[6]/a/img"));

button.click();

//Thread.sleep(3000);

// driver.close();

/\* driver.switchTo().window(newAdwinID1);

System.out.println(driver.getTitle());

Thread.sleep(3000);

driver.close();

\*/

driver.switchTo().window(mainWinID);

System.*out*.println(driver.getTitle());

// Thread.sleep(2000);\*/

WebElement email\_id= driver.findElement(By.*id*("qp"));

email\_id.sendKeys("hi");

Thread.*sleep*(5000);

driver.close();

driver.quit();

}

}

**Action Classes**

Double Click

**publicvoid** f()

{

WebElement web1 = driver.findElement(By.*xpath*("html/body/div[3]/table/tbody/tr[2]/td[1]/div/table/tbody/tr[2]/td[2]/div/table/tbody/tr[2]/td/table/tbody/tr[4]/td[5]"));

web1.click();

WebElement myElemment = driver.findElement(By.*xpath*("html/body/div[3]/table/tbody/tr[2]/td[2]/table/tbody/tr[2]/td[2]/div/div/div/div/table/tbody/tr/td[2]/div/div[53]/div[2]"));

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

action.doubleClick(myElemment);

action.perform();

}

Text selectionke

Drag and drop

Right click done

Slider done

Double click done

Move to element done

clickAndHold

selectmultiple

http://selenium-training-way2automation.blogspot.in/2013/04/magic-of-actions-class-with-webdriver.html

* ButtonReleaseAction - Releasing a held mouse button.
* ClickAction - Equivalent to WebElement.click()
* ClickAndHoldAction - Holding down the left mouse button.
* ContextClickAction - Clicking the mouse button that (usually) brings up the contextual menu.
* DoubleClickAction - double-clicking an element.
* KeyDownAction - Holding down a modifier key.
* KeyUpAction - Releasing a modifier key.
* MoveMouseAction - Moving the mouse from its current location to another element.
* MoveToOffsetAction - Moving the mouse to an offset from an element (The offset could be negative and the element could be the same element that the mouse has just moved to).
* SendKeysAction - Equivalent to WebElement.sendKey(...)

**ScreenShot-**

**public void** screenshot(ITestResult tr){

WebDriver driver=BrowserOpenAndClose.*getInstance*();

**if** ( (driver **instanceof** InternetExplorerDriver) || (driver **instanceof** FirefoxDriver)

|| (driver **instanceof** ChromeDriver)|| (driver **instanceof** OperaDesktopDriver)) {

**try** {

String ScreenshotPath;

// Get the dir path

File folder = **new**File("E:\\Docs");

DateFormat dateFormat = **new**SimpleDateFormat(

"dd\_MMM\_yyyy\_\_hh\_mm\_ssaa");

// get current date time with Date()

Date date = **new**Date();

// To identify the system

//InetAddress ownIP = InetAddress.getLocalHost();

ScreenshotPath = folder.getCanonicalPath() + "\\screenshot\\"+tr.getName()

+ dateFormat.format(date) + "\_" + ".png";

//Count++;

/\*Robot robot = new Robot();

BufferedImage bi = robot.createScreenCapture(new Rectangle(1280,

1024));

ImageIO.write(bi, "png", new File(NewFileNamePath));\*/

count++;

System.*out*.println(ScreenshotPath);

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

FileUtils.*copyFile*(scrFile, **new** File(ScreenshotPath));

ScreenshotPath = "<a href=" + ScreenshotPath + ">ScreenShot"

+count+"</a>";

Reporter.*log*(ScreenshotPath);

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

## Take A Screenshot On The Page

## File snapshot =((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

## Switch between Frames in Java Using Webdriver

## WebElement frameElement = driver.findElement(By.id("id-of-frame"));

## driver.switchTo().frame(frameElement);

## Scroll End of the Page

jsx.executeScript("window.scrollTo(0, document.body.scrollHeight)");

**Scroll Up full window**

jsx.executeScript("window.scrollTo(document.body.scrollHeight,0)");

## 

## vertical + horizontal scroll

1st jsx.executeScript("window.scrollTo(100,document.body.scrollHeight)");

2nd

jsx.executeScript("window.scrollBy(100,500)", "");

**Report**

The report is generated by default under the folder named **testoutput** and can be changed to any other folder by configuring it. These reports consist of certain HTML and XML reports that are TestNG

1. There are two main ways to generate a report with TestNG:

* **Listeners :** For implementing a listener class, the class has to implement the*org.testng.ITestListener* interface. These classes are notified at runtime by TestNG when the test starts, finishes, fails, skips, or passes.
* **Reporters :** For implementing a reporting class, the class has to implement an*org.testng.IReporter* interface. These classes are called when the whole suite run ends. The object containing the information of the whole test run is passed to this class when called.

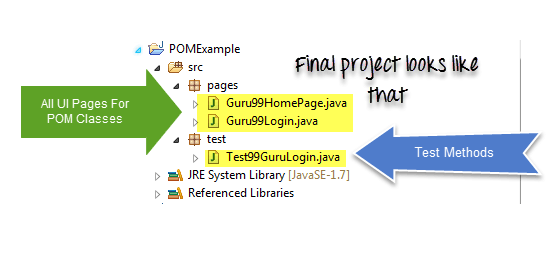
**Why should I use Page Object?**

**Answer**

Page object is a design pattern which distinguishes the code carrying out operations on page and code which carries out tests (assertion/verification). While implementing page object you abstract functioning of a page or part of it in a dedicated “Classs” which is then used by test script to perform actions on page and reach a stage when actual test could be performed.

## What is POM?

* **Page Object Model** is a design pattern to create **Object Repository** for web UI elements.
* Under this model, for each web page in the application, there should be corresponding page class.
* This Page class will find the WebElements of that web page and also contains Page methods which perform operations on those WebElements.
* Name of these methods should be given as per the task they are performing, i.e., if a loader is waiting for the payment gateway to appear, POM method name can be waitForPaymentScreenDisplay().

[](https://cdn.guru99.com/images/AdvanceSelenium/071514_0722_PageObjectM3.png)

## Advantages of POM

1. Page Object Patten says operations and flows in the UI should be separated from verification. This concept makes our code cleaner and easy to understand.
2. The Second benefit is the **object repository is independent of test cases**, so we can use the same object repository for a different purpose with different tools. For example, we can integrate POM with TestNG/JUnit for functional[Testing](https://www.guru99.com/software-testing.html)and at the same time with JBehave/Cucumber for acceptance testing.
3. Code becomes less and optimized because of the reusable page methods in the POM classes.
4. **Methods** get **more realistic names** which can be easily mapped with the operation happening in UI. i.e. if after clicking on the button we land on the home page, the method name will be like 'gotoHomePage()'.

**TestNG XML**

### 3 - Testng.xml

You can invoke TestNG in several different ways:

* With a testng.xml file
* [With ant](http://testng.org/doc/ant.html)
* From the command line

**Group executions**

}

@Test(groups="smoke")

Publicvoid testingMethod1\_1(){

System.out.println("Method - testingMethod1\_1()");

}

@Test(groups="regression")

publicvoid testingMethod4(){

System.out.println("Method - testingMethod4()");

}

}

You can execute the unit test with group â€œmethod1â€ only.

<suite name="Sample Suite">

<test name="testing">

<groups> <run>

<include name="Regression"/>

</run>

</groups>

<classes>

<class name="com.example.group.groupExamples" />

</classes>

</test>

</suite>

* 4.WD It controls the browser from the OS level,**Selenium-WebDriver makes direct calls to the browser using each browser’s native support for automation**

**For those familiar with Selenium-RC, this is quite different from what you are used to. Selenium-RC worked the same way for each supported browser. It ‘injected’ javascript functions into the browser when the browser was loaded and then used its javascript to drive the AUT within the browser. WebDriver does not use this technique. Again, it drives the browser directly using the browser’s built in support for automation.**<http://stackoverflow.com/questions/11535950/difference-between-selenium-rc-and-webdriver>

5,first in rc You first need to launch **a separate application called Selenium Remote Control (RC) Server** before you can start testing

6. The Selenium RC Server **acts as a “middleman” between your Selenium commands and your browser**

**7.** The Selenium RC Server **acts as a “middleman” between your Selenium commands and your browser**

 When you begin testing, Selenium RC Server “injects” a **Javascript program called Selenium Core** into the browser.

 Once injected, Selenium Core will start receiving instructions relayed by the RC Server from your test program.

 When the instructions are received, **Selenium Core will execute them as Javascript commands.**

 The browser will obey the instructions of Selenium Core, and will relay its response to the RC Server.

 The RC Server will receive the response of the browser and then display the results to you.

 RC Server will fetch the next instruction from your test script to repeat the whole cycle.  
Read more at <http://www.guru99.com/introduction-webdriver-comparison-selenium-rc.html#50Gbev3o8V3MfrFq.99>

**Selenium RC cannot support the headless HtmlUnit browser,** HtmlUnit is termed as “headless” because it is an invisible browser – it is GUI-less.

 The only drawbacks of WebDriver are:

** It** **cannot readily support new browsers, but Selenium RC can.**

** It** **does not have a built-in command** **for automatic generation of test results.**

**<!DOCTYPE suite SYSTEM "http://beust.com/testng/testng-1.0.dtd" >**

**<suite name="My test suite">**

**<test name="testing">**

**<classes>**

**<class name="TestNGTest1" />**

**<class name="TestNGTest2" />**

**</classes>**

**</test>**

**</suite>**

**http://softwaretestingbrains.blogspot.in/2013/02/selenium-interview-questions-and.html**

**http://totalinterviewquestions.blogspot.in/2011/10/selenium-rc-latest-interview-questions.html**

**http://seleniumgroup.blogspot.in/p/selenium-ide.html**

**http://www.seleniumguide.com/**

**http://qaknowledgesharing.blogspot.in/2012/06/selenium-ide-interview-questions-and.html**

**http://qaknowledgesharing.blogspot.in/2012/06/selenium-ide-interview-questions-and.html**

[**http://www.scribd.com/doc/81850271/Selenium-Tester-Sample-Resume**](http://www.scribd.com/doc/81850271/Selenium-Tester-Sample-Resume)

**TESTNG**

#### Dependencies

#### Preserver Order in Testng. If you want your classes / methods to be run in an unpredictable order, then we should go for preserve-order attribute in testng. InTestNg bydefault the preserve-order attribute will be set to 'true', this means, TestNG will run your tests in the order they are found in the XML file

TestNG allows you to specify dependencies either with annotations or in XML.

##### **5.7.1 - Dependencies with annotations**

You can use the attributes dependsOnMethods or dependsOnGroups, found on the @Test annotation.

There are two kinds of dependencies:

**Hard dependencies**

Let us say test1 method depends on test2 method.

In this type of dependency, if test2 fails, test1 is not executed and marked as Skipped. This is the default dependency.

**Soft dependencies**

In this type of dependency, even if test2 fails, test1 is executed. This is achieved by adding alwaysRun=true as shown in below example.

@Test(dependsOnMethods = "org.softpost.Class2.test2", alwaysRun = true)

**public** void test1(){

**System**.out.println("Test1 from Class1");

}

***6.2 - Parameters with DataProviders***

Specifying parameters in testng.xml might not be sufficient if you need to pass complex parameters, or parameters that need to be created from Java (complex objects, objects read from a property file or a database, etc...). In this case, you can use a Data Provider to supply the values you need to test.  A Data Provider is a method on your class that returns an array of array of objects.  This method is annotated with @DataProvider:

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

|  |
| --- |
| @DataProvider(name = "test1") |
| public Object[][] createData1() { | |

|  |
| --- |
| return new Object[][] { |
| { "Cedric", new Integer(36) }, | |

|  |  |
| --- | --- |
| { "Anne", new Integer(37)}, | |
| }; |

|  |  |
| --- | --- |
| } | |
|  |

|  |  |
| --- | --- |
| //This test method declares that its data should be supplied by the Data Provider | |
| //named "test1" |

|  |
| --- |
| @Test(dataProvider = "test1") |
| public void verifyData1(String n1, Integer n2) { | |

|  |  |
| --- | --- |
| System.out.println(n1 + " " + n2); | |
| } |

will print

|  |  |
| --- | --- |
| Cedric 36 | |
| Anne 37 |

A @Test method specifies its Data Provider with the dataProvider attribute.  This name must correspond to a method on the same class annotated with @DataProvider(name="...") with a matching name.

By default, the data provider will be looked for in the current test class or one of its base classes. If you want to put your data provider in a different class, it needs to be a static method and you specify the class where it can be found in thedataProviderClass attribute:

|  |
| --- |
| public class StaticProvider { |
| @DataProvider(name = "create") | |

|  |  |
| --- | --- |
| public static Object[][] createData() { | |
| return new Object[][] { |

|  |  |
| --- | --- |
| new Object[] { new Integer(42) } | |
| } |

|  |  |
| --- | --- |
| } | |
| } |

|  |
| --- |
|  |
| public class MyTest { | |

|  |  |
| --- | --- |
| @Test(dataProvider = "create", dataProviderClass = StaticProvider.class) | |
| public void test(Integer n) { |

|  |  |
| --- | --- |
| // ... | |
| } |

|  |
| --- |
| } |

The Data Provider method can return one of the following two types:

* An array of array of objects (Object[][]) where the first dimension's size is the number of times the test method will be invoked and the second dimension size contains an array of objects that must be compatible with the parameter types of the test method. This is the cast illustrated by the example above.
* An Iterator<Object[]>. The only difference with Object[][] is that an Iterator lets you create your test data lazily. TestNG will invoke the iterator and then the test method with the parameters returned by this iterator one by one. This is particularly useful if you have a lot of parameter sets to pass to the method and you don't want to create all of them upfront.

Here is an example of this feature:

|  |
| --- |
| @DataProvider(name = "test1") |
| public Iterator<Object[]> createData() { | |

|  |  |
| --- | --- |
| return new MyIterator(DATA); | |
| } |

If you declare your @DataProvider as taking a java.lang.reflect.Method as first parameter, TestNG will pass the current test method for this first parameter. This is particularly useful when several test methods use the same@DataProvider and you want it to return different values depending on which test method it is supplying data for.

For example, the following code prints the name of the test method inside its @DataProvider:

|  |
| --- |
| @DataProvider(name = "dp") |
| public Object[][] createData(Method m) { | |

|  |  |
| --- | --- |
| System.out.println(m.getName());  // print test method name | |
|  | |
| return new Object[][] { new Object[] { "Cedric" }}; |

|  |  |
| --- | --- |
| } | |
|  |

|  |
| --- |
| @Test(dataProvider = "dp") |
| public void test1(String s) { | |

|  |  |
| --- | --- |
| } | |
|  |

|  |
| --- |
| @Test(dataProvider = "dp") |
| public void test2(String s) { | |

|  |
| --- |
| } |

and will therefore display:

|  |
| --- |
| test1 |
| test2 |

Data providers can run in parallel with the attribute parallel:

|  |  |
| --- | --- |
| @DataProvider(parallel = true) | |
| // ... |

Parallel data providers running from an XML file share the same pool of threads, which has a size of 10 by default. You can modify this value in the <suite> tag of your XML file:

|  |  |
| --- | --- |
| <suite name="Suite1" data-provider-thread-count="20" > | |
| ... |

If you want to run a few specific data providers in a different thread pool, you need to run them from a different XML file.

*5.6.3 - Parameters in reports*

Parameters used to invoke your test methods are shown in the HTML reports generated by TestNG. Here is an example:

**.10.2 - Parallel tests, classes and methods**

The *parallel* attribute on the <suite> tag can take one of following values:

|  |  |
| --- | --- |
| <suite name="My suite" parallel="methods" thread-count="5"> | |
| <suite name="My suite" parallel="tests" thread-count="5"> |

|  |
| --- |
| <suite name="My suite" parallel="classes" thread-count="5"> |

[view source](http://testng.org/doc/documentation-main.html#viewSource)

[print](http://testng.org/doc/documentation-main.html#printSource)[?](http://testng.org/doc/documentation-main.html#about)

|  |
| --- |
| <suite name="My suite" parallel="instances" thread-count="5"> |

* **parallel="methods"**: TestNG will run all your test methods in separate threads. Dependent methods will also run in separate threads but they will respect the order that you specified.
* **parallel="tests"**: TestNG will run all the methods in the same <test> tag in the same thread, but each <test> tag will be in a separate thread. This allows you to group all your classes that are not thread safe in the same <test> and guarantee they will all run in the same thread while taking advantage of TestNG using as many threads as possible to run your tests.
* **parallel="classes"**: TestNG will run all the methods in the same class in the same thread, but each class will be run in a separate thread.
* **parallel="instances"**: TestNG will run all the methods in the same instance in the same thread, but two methods on two different instances will be running in different threads.

### Assert vs. Verify

Here’s the trade-off. If you use an assert, the test will stop at that point and not run any subsequent checks. Sometimes, perhaps often, that is what you want. If the test fails you will immediately know the test did not pass. Test engines such as TestNG and JUnit have plugins for commonly used development environments (Chap 5) which conveniently flag these tests as failed tests. The advantage: you have an immediate visual of whether the checks passed. The disadvantage: when a check does fail, there are other checks which were never performed, so you have no information on their status.

**preserve-order** - use to run method ,classes sequentially

Listener - Invoked each time before a test will be invoked.

Listeners are basically those entities which are tuned into one or more events that may arise.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Requirement Analysis
* Test Planning
* Test Case Development
* Environment Setup
* Test Execution
* Test Cycle Closure

**What is TestNG?**

**A:** TestNG is an open source automated testing framework; where NG of TestNG means Next Generation. TestNG is similar to JUnit (especially JUnit 4), but its not a JUnit extension. Its inspired by JUnit. It is designed to be better than JUnit, especially when testing integrated classes.

**What are the features of TestNG?**

**A:** Features of TestNG are:

* Annotations.
* TestNG uses more Java and OO features.
* Supports testing integrated classes (e.g., by default, no need to create a new test class instance for every test method).
* Separate compile-time test code from run-time configuration/data info.
* Flexible runtime configuration.
* Introduces ‘test groups’. Once you have compiled your tests, you can just ask TestNG to run all the "front-end" tests, or "fast", "slow", "database", etc...
* Supports Dependent test methods, parallel testing, load testing, partial failure.
* Flexible plug-in API.
* Support for multi threaded testing.

**Order of Sequence in Annotations**

<http://www.tutorialspoint.com/testng/testng_execution_procedure.htm>

beforeSuite

beforeTest

beforeClass

beforeMethod. Execute the method just before executing any test method in the class.

test case 1

afterMethod Execute the method after executing every test method in the class

beforeMethod

test case 2

afterMethod

afterClass

afterTest

afterSuite

**What are the different ways in which TestNG can be invoked?**

**A:** You can invoke TestNG in several different ways:

* Using Eclipse
* With ant
* From the command line

**Verbose in testNG.xml**

<http://seleniumone-by-arun.blogspot.in/2013/05/158-understanding-usage-of-verbose.html>

It can be 1 to 10 value , it defines the log file detail in eclipse 1 is less detail 10 is large details

**How to you specify a group in testng.xml?**

 sophisticated groupings of test methods.  Not only can you declare that methods belong to groups, but you can also specify groups that contain other groups.

|  |
| --- |
| public class Test1 { |

|  |
| --- |
| @Test(groups = { "functest", "checkintest" }) |

|  |
| --- |
| public void testMethod1() { |

|  |
| --- |
| } |

|  |
| --- |
|  |

|  |
| --- |
| @Test(groups = {"functest", "checkintest"} ) |

|  |
| --- |
| public void testMethod2() { |

|  |
| --- |
| } |

|  |
| --- |
|  |

|  |
| --- |
| @Test(groups = { "functest" }) |

|  |
| --- |
| public void testMethod3() { |

|  |
| --- |
| } |

|  |
| --- |
| } |

**Invoking TestNG with**

|  |  |
| --- | --- |
| <test name="Test1"> | |
| <groups> |

|  |
| --- |
| <run> |
| <include name="functest"/> | |

|  |
| --- |
| </run> |
| </groups> | |

|  |
| --- |
| <classes> |
| <class name="example1.Test1"/> | |

|  |  |
| --- | --- |
| </classes> | |
| </test> |
| **: What is parametric testing?**  **A:** In most cases, you'll come across a scenario where the business logic requires a hugely varying number of tests. *Parameterized tests* allow developers to run the same test over and over again using different values.  TestNG lets you pass parameters directly to your test methods in two different ways:   * With testng.xml * With Data Providers   **Q: What are different ways in which you can generate the reports of TestNg results?**  **A:** There are two main ways to generate a report with TestNG:   * **Listeners:** For implementing a listener class, the class has to implement the*org.testng.ITestListener* interface. These classes are notified at runtime by TestNG when the test starts, finishes, fails, skips, or passes. * **Reporters:** For implementing a reporting class, the class has to implement an*org.testng.IReporter* interface. These classes are called when the whole suite run ends. The object containing the information of the whole test run is passed to this class when called   Step 7: Verify the Result  Compile the class using **javac** compiler as follows:  C:\TestNG\_WORKSPACE>javac TestNGSimpleTest.java  Now, invoke the testng.xml to see the result:  C:\TestNG\_WORKSPACE>java -cp "C:\TestNG\_WORKSPACE" org.testng.TestNG testng.xml   |  |  | | --- | --- | | **DataProvider** | Marks a method as supplying data for a test method. The annotated method must return an Object[ ][ ] where each Object[ ] can be assigned the parameter list of the test method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation. |  1. Core- All predefined events like –type, click,selct   **publicvoid** click(By locator){  driver.findElement(locator).click();  }   1. Elements 2. Screenshot 3. Test 4. DataProvider 5. Exelhandler |

Data provider – take input from excel

@Data Provider

**publicstatic** Object[][] ValidUsernameAndPasswordProvider(ITestContext context) **throws** Exception{

String xlsloc=context.getCurrentXmlTest().getParameter("loginXlsLoc");

String sheet=context.getCurrentXmlTest().getParameter("SheetName");

ExcelHandler excel=**new**ExcelHandler(**new** File(System.*getProperty*("user.dir")+xlsloc));

excel.selectSheet(sheet);

String username=excel.getColumn(4).get(11).getContents();

String password=excel.getColumn(5).get(11).getContents();

**returnnew** Object[][]{{username,password}};

}

Use in @test

@Test(description="1. Type valid credentials and hit login button",dataProviderClass=DataProviderForLogin.**class**,dataProvider="ValidUsernameAndPasswordProvider")

**publicvoid** Verify\_ValidUsernameAndPassword(String username,String password) **throws** Exception{

LoginPage loginPage=**new**LoginPage(*driver*);

HomePageElements hpe=**new**HomePageElements();

HomePage homePage=loginPage.SignIn(username,password);

System.*out*.println(*driver*.findElement(hpe.getsuccesstest()).getText());

**assert** homePage.textEqualTo(hpe.getsuccesstest(), "Hello Sunaina"):"Expected - Login should be successfully";

}

}

**CheckElement Present**

I used java print statements for easy understanding.

1. To check Element Present:
2. if(driver.findElements(By.xpath("value")).size() != 0){
3. System.out.println("Element is Present");
4. }else{
5. System.out.println("Element is Absent");

}

Or

if(driver.findElement(By.xpath("value"))!= null){

System.out.println("Element is Present");

}else{

System.out.println("Element is Absent");

}

1. To check Visible:
2. if( driver.findElement(By.cssSelector("a > font")).isDisplayed()){
3. System.out.println("Element is Visible");
4. }else{
5. System.out.println("Element is InVisible");

}

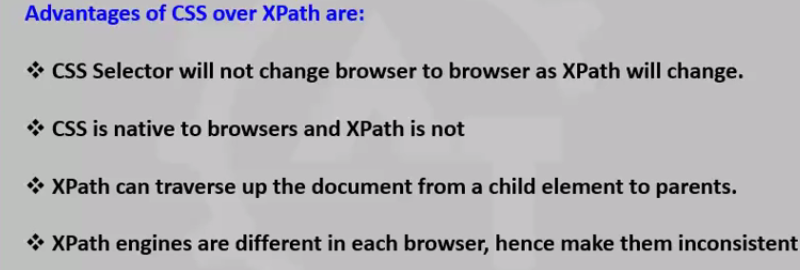
1. To check Enable:
2. if( driver.findElement(By.cssSelector("a > font")).isEnabled()){
3. System.out.println("Element is Enable");
4. }else{
5. System.out.println("Element is Disabled");

}

1. To check text present
2. if(driver.getPageSource().contains("Text to check")){
3. System.out.println("Text is present");
4. }else{
5. System.out.println("Text is absent");

}

**Css is faster than xpath**



**Xpath function $x**

wikipedia

$x("//input[@name='q']")

$x("//input[starts-with(@id, 'search')]")

$x("//input[substring(@id, 2,5)='earch']")

$x("//title[normalize-space(.)='selenium - Google Search']")

$x("//input[contains(@id, 'hIn')]")

$x("//input[contains(@id, 'hIn')]/following-sibling::select")

Even preceding-sibling exist just to one backward same element

Parent one

$x("//input[contains(@id, 'hIn')]/parent::fieldset")

Nth-child[1] vs nth-of-type[1]

For immediate parent

$x("//input[contains(@id, 'hIn')]/..")

Parent to parent

$x("//input[contains(@id, 'hIn')]//..//..//..//..//..//..")

**$$("input.inputtext.\_58mg.\_5dba.\_2ph-")**

**$$(input#u\_0\_3)**

**$$("input.inputtext.\_58mg.\_5dba.\_2ph-")**

**$$("input[name^='first']") --initial name pattern**

**$$("input[name$='name']") last name patterent**

**$$("input[id\*='\_0\_']") - contains**

**You can't do this with CSS selectors, because there is no such keyword :contains() in CSS. It was a proposal that was abandoned years ago.**

**for 2 conditions**

**$$("input[name^='first'][name='firstname']")**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

https://developer.mozilla.org/en-US/docs/Web/CSS/:nth-of-type

<p><code>span:nth-child(2n+1)</code>, <em>with</em> an <code>&lt;em&gt;</code>

inside the child elements. Children 1, 5, and 7 are selected. 3 is used in the

counting because it is a child, but it isn't selected because it isn't a

<code>&lt;span&gt;</code>.</p>

<div class="second">

<span>This span is selected!</span>

<span>This span is not. :(</span>

<em>This one is an em.</em>

<span>What about this?</span>

<span>And this one?</span>

<span>Another example</span>

<span>Yet another example</span>

<span>Aaaaand another</span>

</div>

<p><code>span:nth-of-type(2n+1)</code>, <em>with</em> an <code>&lt;em&gt;</code>

inside the child elements. Children 1, 4, 6, and 8 are selected. 3 isn't

used in the counting or selected because it is an <code>&lt;em&gt;</code>,

not a <code>&lt;span&gt;</code>, and <code>nth-of-type</code> only selects

children of that type. The <code>&lt;em&gt;</code> is completely skipped

over and ignored.</p>

<div class="third">

<span>This span is selected!</span>

<span>This span is not. :(</span>

<em>This one is an em.</em>

<span>What about this?</span>

<span>And this one?</span>

<span>Another example</span>

<span>Yet another example</span>

<span>Aaaaand another</span>

</div>

Example selectors

tr:nth-child(2n+1)

Represents the odd rows of an HTML table.

tr:nth-child(odd)

Represents the odd rows of an HTML table.

tr:nth-child(2n)

Represents the even rows of an HTML table.

tr:nth-child(even)

Represents the even rows of an HTML table.

span:nth-child(0n+1)

Represents a span element which is the first child of its parent; this is the same as the [:first-child](https://developer.mozilla.org/en-US/docs/Web/CSS/:first-child) selector.

span:nth-child(1)

Equivalent to the above.

span:nth-child(-n+3)

Matches if the element is one of the first three children of its parent and also a span.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

<ul>

<li class="active">

<a href="#">

<i class="fa fa-home"></i><br>

<span class="title">Home</span>

</a>

</li>

<li>

<a href="#">

<i class="fa fa-rss-square"></i><br>

<span class="title">Posts</span>

</a>

</li>

<li>

<a href="#">

<i class="fa fa-calendar"></i><br>

<span class="title">Events</span>

</a>

</li>

<li>

<a href="#">

<i class="fa fa-bar-chart-o"></i><br>

<span class="title">My Activity</span>

</a>

</li>

<li>

<a href="#">

<i class="fa fa-edit"></i><br>

<span class="title">Assessments</span>

</a>

</li>

</ul>

driver.findElement(By.cssSelector("ul > li:nth-child(1)")); >> home

driver.findElement(By.cssSelector("ul > li:nth-child(2)")); >> posts

driver.findElement(By.cssSelector("ul > li:nth-child(3)")); >> events

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CSS Selector Rule - :nth-child(n)**  
Example -   
For the Sample HTML below-

<ul id="testingTypes">

<li>Automation Testing</li>

<li>Performance Testing</li>

<li>Manual Testing</li>

</ul>

CSSLocator-#testingTypes li: nth-child(2)

Description - '#testingTypes li:nth-child(2)' will select the element with id 'testingType' and then locate the 2nd child of type li i.e. 'Performance Testing' list item.

**CSSSelectorRule-locator1+locator2**  
Example -   
For the Sample HTML below-

<ul id="testingTypes">

<li id="automation">Automation Testing</li>

<li>Performance Testing</li>

<li>Manual Testing</li>

</ul>

CSSLocator->li#automation+li  
Description - 'li#automation + li' will first go to li element with id 'automation' and then select its adjacent li i.e. 'Performance Testing' list item.

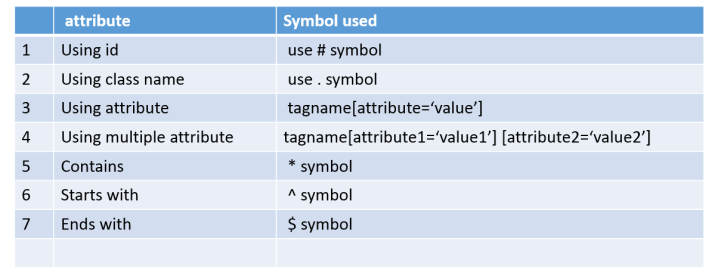
**parentLocator>childLocator**  
Example -   
For the Sample HTML below-

<div id="buttonDiv" class="small">

<button id="submitButton1" type="button" class="btn">Submit</button>

</div>

CSS Locator - div#buttonDiv>button



# [Identify dynamic element in selenium](http://selenium-suresh.blogspot.com/2012/08/identify-dynamic-element-in-selenium.html)

**How to identify dynamic element in selenium**  
  
Many web sites create dynamic element on their web pages where Ids of the elements gets generated dynamically. Each time id gets generated differently. So to handle this situation we use some JavaScript functions.  
**starts-with**   
if your dynamic element's ids have the format  where button id="continue-12345"  where 12345 is a dynamic number you could use the following  
**XPath:** //button[starts-with(@id, 'continue-')]   
**contains**   
and ‘business’ classes using the following

Sometimes an element gets identfied by a value that could be surrounded by other text, then contains function can be used.

To demonstrate, the element can be located based on the ‘suggest’ class without having to couple it with the ‘top’  
**XPath:** //input[contains(@class, 'suggest')].

//elementname[@attribute=’value’]

Regular expression //\*[

Javascript functions

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

Typesoframework

<http://testautomationarchives.blogspot.in/2013/09/automation-frameworks-with-example.html>

**This will help you to check whether required text is there in webpage or not.**

driver.getPageSource().contains("Text which you looking for");

Thus, when you try to mix implicit and explicit waits, you've strayed into "undefined behavior". You might be able to figure out what the rules of that behavior are, but they'll be subject to change as the implementation details of the drivers change. So don't do it.

The best practice is to set implicitlyWait() at the beginning of each test, and use WebDriverWait() for waiting an element, or AJAX element to load.

However, implicitlyWait() and WebDriverWait() do not work well together in the same test. You would have to nullify implicitlyWait() before calling WebDriverWait because implicitlyWait() also sets the "driver.findElement()" wait time.

Whenever you are using WebDriverWait() with implicitlyWait() already set some initial value, follow the steps -

1. nullifying implicitlyWait()
2. executing WebDriverWait (), and return element
3. reset implicitlyWait() again

Example Java code -

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

WebDriverWait wait = new WebDriverWait(driver, timeOutInSeconds);

element = wait.until(ExpectedConditions.visibilityOfElementLocated(by));

driver.manage().timeouts().implicitlyWait(DEFAULT\_WAIT\_4\_PAGE, TimeUnit.SECONDS)

1. When we use x path manually-

When both id and name is same or doest not exist and element is generated dynamilly and id changes dynamilly then we use xptah manually

<http://stackoverflow.com/questions/17985733/how-to-handle-dynamically-changing-ids-with-similar-starting-name-using-webdriv>

# [How to handle dynamically changing id's with similar starting name using Webdriver](http://stackoverflow.com/questions/17985733/how-to-handle-dynamically-changing-ids-with-similar-starting-name-using-webdriv)

|  |  |
| --- | --- |
| vote[favorite](http://stackoverflow.com/questions/17985733/how-to-handle-dynamically-changing-ids-with-similar-starting-name-using-webdriv) | I am automating the test for web application. I have a scenario for creating an admin, for which i have to enter the name, email address and phone number text boxes. But ids of this text boxes are dynamic.  userName, id='oe-field-input-41'  Email, id='oe-field-input-42'  phone number, id='oe-field-input-43'  First Query: The numbers in the ids are dynamic, it keep changes I tired to use the xpath for handling the dynamic value.  xpath =//\*[starts-with(@id,'oe-field-input-')]  In this it enter the text into first text box successfully  Second Query: I am not able use the same xpath for next two text boxes, as it enters the email and phone number into name field only  Please help me to resolve this dynamic value handling.  Edited: added the html code,  <tableclass="oe\_form\_group "cellspacing="0"cellpadding="0"border="0">  <tbody>  <trclass="oe\_form\_group\_row">  <tdclass="oe\_form\_group\_cell oe\_form\_group\_cell\_label"width="1%"colspan="1">  <tdclass="oe\_form\_group\_cell"width="99%"colspan="1">  <spanclass="oe\_form\_field oe\_form\_field\_many2one oe\_form\_field\_with\_button">  <aclass="oe\_m2o\_cm\_button oe\_e"tabindex="-1"href="#"draggable="false"style="display: inline;">/</a>  <div>  </span>  </td>  </tr>  <trclass="oe\_form\_group\_row">  <tdclass="oe\_form\_group\_cell oe\_form\_group\_cell\_label"width="1%"colspan="1">  <tdclass="oe\_form\_group\_cell"width="99%"colspan="1">  <spanclass="oe\_form\_field oe\_form\_field\_email">  <div>  <inputid="oe-field-input-35"type="text"maxlength="240">  </div>  </span>  </td>  </tr>  <trclass="oe\_form\_group\_row">  <tdclass="oe\_form\_group\_cell oe\_form\_group\_cell\_label"width="1%"colspan="1">  <tdclass="oe\_form\_group\_cell"width="99%"colspan="1">  <spanclass="oe\_form\_field oe\_form\_field\_char">  <inputid="oe-field-input-36"type="text"maxlength="32">  </span>  </td>  </tr>  <trclass="oe\_form\_group\_row">  <tdclass="oe\_form\_group\_cell oe\_form\_group\_cell\_label"width="1%"colspan="1">  <tdclass="oe\_form\_group\_cell"width="99%"colspan="1">  <spanclass="oe\_form\_field oe\_form\_field\_char">  <inputid="oe-field-input-37"type="text"maxlength="32">  </span>  </td>  </tr>  <trclass="oe\_form\_group\_row">  </tbody> |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | Can you please share the html. Should be very easy once I see the html. –  [Vinay](http://stackoverflow.com/users/2147846/vinay) [Aug 1 '13 at 5:34](http://stackoverflow.com/questions/17985733/how-to-handle-dynamically-changing-ids-with-similar-starting-name-using-webdriv#comment26295815_17985733) | | |  |  | | --- | --- | |  |  | | @Vinay, i have edited the post and added the html. Please provide your input. –  [mdashu](http://stackoverflow.com/users/1692398/mdashu) [Aug 1 '13 at 5:48](http://stackoverflow.com/questions/17985733/how-to-handle-dynamically-changing-ids-with-similar-starting-name-using-webdriv#comment26296086_17985733) | | |  |  | | --- | --- | |  |  | | the html you have shared does not contain any names to the text field. Is this the complete html or the text fields do not have any names at all? –  [Vinay](http://stackoverflow.com/users/2147846/vinay) [Aug 1 '13 at 5:53](http://stackoverflow.com/questions/17985733/how-to-handle-dynamically-changing-ids-with-similar-starting-name-using-webdriv#comment26296177_17985733) | | |  |  | | --- | --- | |  |  | | @Vinay, It has the id values for three text boxs. <input id="oe-field-input-35" type="text" maxlength="240">, <input id="oe-field-input-36" type="text" maxlength="32"> and <input id="oe-field-input-37" type="text" maxlength="32"> –  [mdashu](http://stackoverflow.com/users/1692398/mdashu) [Aug 1 '13 at 5:58](http://stackoverflow.com/questions/17985733/how-to-handle-dynamically-changing-ids-with-similar-starting-name-using-webdriv#comment26296267_17985733) |   add comment |

## 5 Answers

[active](http://stackoverflow.com/questions/17985733/how-to-handle-dynamically-changing-ids-with-similar-starting-name-using-webdriv?answertab=active#tab-top)[oldest](http://stackoverflow.com/questions/17985733/how-to-handle-dynamically-changing-ids-with-similar-starting-name-using-webdriv?answertab=oldest#tab-top)[votes](http://stackoverflow.com/questions/17985733/how-to-handle-dynamically-changing-ids-with-similar-starting-name-using-webdriv?answertab=votes#tab-top)

|  |  |  |  |
| --- | --- | --- | --- |
| up vote0down vote | The problem here is, that your XPath does the correct selection, but Selenium will always pick the first one if multiple results are returned for your query.  You can select each of the input fields directly by using:  //input[1]  //input[2]  //input[3]  If there are other input fields, you can tighten your selection by selecting only input nodes with oe-field-input in their id attribute like this:  //input[starts-with(@id,'oe-field-input-')][1]  //input[starts-with(@id,'oe-field-input-')][2]  //input[starts-with(@id,'oe-field-input-')][3]   |  |  | | --- | --- | |  |  | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | It's better to verify before posting an answer. The code you have suggested does not work. –  [Vinay](http://stackoverflow.com/users/2147846/vinay) [Aug 1 '13 at 6:17](http://stackoverflow.com/questions/17985733/how-to-handle-dynamically-changing-ids-with-similar-starting-name-using-webdriv#comment26296679_17986383) |   add comment |

|  |  |  |  |
| --- | --- | --- | --- |
| up vote0down vote | Use the following xpath works like a charm. Although I don't recommend this kind of an xpath. Since we don't have text against the text box no other choice.  //div/input[contains(@id, 'oe-field-input')] - First text box  //tr[@class = 'oe\_form\_group\_row'][2]//input - Second text box  //tr[@class = 'oe\_form\_group\_row'][3]//input - Third text box   |  |  | | --- | --- | |  |  | |
|  |  |
| up vote0down vote | You can use below XPATH.  //tr[@class = 'oe\_form\_group\_row'][2]//input for First Text box  //tr[@class = 'oe\_form\_group\_row'][3]//input for Second Text box  //tr[@class = 'oe\_form\_group\_row'][4]//input for Third text box.  I have tested avove xpath.  But the better way if you have development access then ask developers to make is standaralized and recommand tags like "name" , "value", or attach text e.g. Email:, Password. So you can use these in your xpath.   |  |  | | --- | --- | |  |  | |
|  |  |

|  |  |
| --- | --- |
| up vote0down vote | You do have some classes which are good for identification, e.g. oe\_form\_field\_email, oe\_form\_field\_char. It's a little complicated to use them because they're not on the input fields themselves, and the second one is not unique; but it's quite possible:  .//span[contains(@class, 'oe\_form\_field\_email')]//input  That is an xpath which identifies the Email field as being the input which is a descendant of a spanwith the oe\_form\_field\_email class. You could also use the same logic in a css selector like this, more efficiently:  span.oe\_form\_field\_email input  For the two other fields, there is no unique class which can tell them apart so you're going to have to rely on the order (I'm assuming username comes before phone number), and that means you have to use xpaths:  (//tr//span[contains(@class, 'oe\_form\_field\_char')])[1]//input  (//tr//span[contains(@class, 'oe\_form\_field\_char')])[2]//input |

3 text boxes have same id name then how to access

Use index basesd xpath manually

3. Where do we put implicit time

5. how to click second dynamic id

IMP-<http://software-testing-tutorials-automation.blogspot.in/search/label/WebDriver>

**Desire Capability**

Capabilities: Describes a series of key/value pairs that encapsulate aspects of a browser.

Basically, the DesiredCapabilities help to set properties for the WebDriver. A typical usecase would be to set the path for the FirefoxDriver if your local installation doesn't correspond to the default settings.

1. It is a class under org.openqa.selenium.remote.DesiredCapabilities package.

2. It gives facility to set the properties of browser. Such as to set

BrowserName, Platform,Version of Browser.

3. Mostly DesiredCapabilities class used when do we used Selenium Grid.

4. we have to execute mutiple TestCases on multiple Systems with different

browser with Different version and Different Operating System.

Example:

WebDriver driver;

String baseUrl , nodeUrl;

baseUrl = "https://www.facebook.com";

nodeUrl = "http://192.168.10.21:5568/wd/hub";

DesiredCapabilities capability = DesiredCapabilities.firefox();

capability.setBrowserName("firefox");

capability.setPlatform(Platform.WIN8\_1);

driver = new RemoteWebDriver(new URL(nodeUrl),capability);

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

driver.manage().timeouts().implicitlyWait(2, TimeUnit.MINUTES);

|  |  |
| --- | --- |
| [share](http://stackoverflow.com/a/29314353)[improve this answer](http://stackoverflow.com/posts/29314353/edit) |  |

Capabilities: Describes a series of key/value pairs that encapsulate aspects of a browser.

1) Desired Capabilities can set flags/options on a browser when you create it;Capabilities are options that you can use to customize and configure a browser

Basically, the DesiredCapabilities help to set properties for the WebDriver. A typical usecase would be to set the path for the FirefoxDriver if your local installation doesn't correspond to the default settings.

How To Enable/Disable Textbox In Selenium WebDriver On The Fly

@Test

public void test () throws BiffException, IOException, InterruptedException

{

boolean fbefore = driver.findElement(By.xpath("//input[@name='fname']")).isEnabled();

System.out.print("\nBefore : First Name Text box enabled status is : "+fbefore);

boolean lbefore = driver.findElement(By.xpath("//input[@name='lname']")).isEnabled();

System.out.print("\nBefore : Last Name Text box enabled status is : "+lbefore);

Thread.sleep(2000);

**//To disable First Name text box**

JavascriptExecutor javascript = (JavascriptExecutor) driver;

String todisable = "document.getElementsByName('fname')[0].setAttribute('disabled', '');";

javascript.executeScript(todisable);

Thread.sleep(2000);

**//To enable Last Name text box**

String toenable = "document.getElementsByName('lname')[0].removeAttribute('disabled');";

javascript.executeScript(toenable);

Thread.sleep(3000);

boolean fafter = driver.findElement(By.xpath("//input[@name='fname']")).isEnabled();

System.out.print("\nAfter : First Name Text box enabled status is : "+fafter);

boolean lafter = driver.findElement(By.xpath("//input[@name='lname']")).isEnabled();

System.out.print("\nAfter : Last Name Text box enabled status is : "+lafter);

}

Executing javascript in selenium webdriver to get page title with example

Test

Public void test ()

{

JavascriptExecutor javascript = (JavascriptExecutor) driver;

**//Get current page title**

String pagetitle=(String)javascript.executeScript("return document.title");

System.out.println("My Page Title Is : "+pagetitle);

**//Get current page URL**

String CurrentURL = driver.getCurrentUrl();

System.out.println("My Current URL Is : "+CurrentURL);

}

Get Domain Name Using JavascriptExecutor

@Test

Public void test ()

{

String CurrentURL = driver.getCurrentUrl();

System.out.println("My Current URL Is: "+CurrentURL);

**//Get and store domain name in variable**

JavascriptExecutor javascript = (JavascriptExecutor) driver;

String DomainUsingJS=(String)javascript.executeScript("return document.domain");

System.out.println("My Current URL Is : "+DomainUsingJS);

}

**javascriptExecutor to generate alert in selenium webdriver**

@Test

public void test () throws InterruptedException

{

**//Generating Alert Using Javascript Executor**

JavascriptExecutor javascript = (JavascriptExecutor) driver;

javascript.executeScript("alert('Test Case Execution Is started Now..');");

Thread.Sleep(2000);

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

**//Wait for page title**

WebDriverWait wait = new WebDriverWait(driver, 15);

wait.until(ExpectedConditions.titleContains(": MyTest"));

How To Navigate URL, Forward and Backward With Example

**driver.navigate().back()**

Select mydrpdwn = new Select(driver.findElement(By.id("Carlist")));

mydrpdwn.selectByVisibleText("Audi");

Capture Screenshot With Example

@Test

public void test () throws InterruptedException, IOException

{

**//Capture entire page screenshot and then store it to destination drive**

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

// This getScreenshotAs()method takes argument of type OutputType.File or OutputType.BASE64 or Output.BYTES. So that it could return captured screenshot in File type, or Base 64 string type or in raw bytes.

**The**[**WebDriver**](http://selenium.googlecode.com/git/docs/api/java/org/openqa/selenium/WebDriver.html)**interface does not contain the getScreenshotAs() method, because it is possible to have a webdriver unable of taking screenshots - for example the in-memory drivers that don't render the page at all, like HtmlUnitDriver.**

**In order to have the method, the driver must implement the**[**TakesScreenshot**](http://selenium.googlecode.com/git/docs/api/java/org/openqa/selenium/TakesScreenshot.html)**interface which makes it capable to ... well ... take screenshots.**

**Therefore, you must tell the program somehow that you want to take a screenshot and that you are absolutely sure you can do so. That's what the (TakesScreenshot)driver part is for. In Java, it's called**[**casting**](https://www.google.com/search?q=Java+cast)**and it literally translates to *"I know that this driver instance is able to take a screenshot, please cast it to TakesScreenshot type."***

**If your cast succeeds, everything is fine and the driver object will be cast at run-time to an instance of TakesScreenshot. If your cast fails, however, you'll get a**[**ClassCastExcepion**](http://docs.oracle.com/javase/7/docs/api/java/lang/ClassCastException.html)**at run-time.**

So this would look like this

FileUtils.copyFile(screenshot, new File("D:\\screenshot.jpg"));

System.out.print("Screenshot is captured and stored in your D: Drive");

}

Generating Mouse Hover Event On Main Menu To Click On Sub Menu

Actions actions = new Actions(driver);

WebElement moveonmenu = driver.findElement(By.xpath("//div[@id='menu1']/div"));

actions.moveToElement(moveonmenu).moveToElement(driver.findElement(By.xpath("//div[@id='menu1choices']/a"))).click().perform();

WebDriverWait wait = new WebDriverWait(driver, 15);

wait.until(ExpectedConditions.titleContains("Google"));

How to wait till element visible or appear or present on page

Test

public void test () throws InterruptedException, IOException

{

**//To wait for element visible**

WebDriverWait wait = new WebDriverWait(driver, 15);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//input[@id='text3']")));

driver.findElement(By.xpath("//input[@id='text3']")).sendKeys("Text box is visible now");

System.out.print("Text box text3 is now visible");

}

Extract All Links From Web Page Using Selenium WebDriver

@Test

public void test () throws InterruptedException

{

try {

List<WebElement> no = driver.findElements(By.tagName("a"));

int nooflinks = no.size();

System.out.println(nooflinks);

for (WebElement pagelink : no)

{

String linktext = pagelink.getText();

String link = pagelink.getAttribute("href");

System.out.println(linktext+" ->");

System.out.println(link);

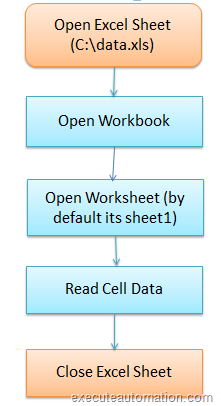
}

}catch (Exception e){

System.out.println("error "+e);

}

}



**whats the difference between Junit and TestNG**

Also using TestNG your selenium testcase execution can be done in parallel.

1. In Junit we have to declare @BeforeClass and @AfterClass which is a constraint where as in TestNG there is no constraint like this.
2. **TestNG supports parameterization for objects.**
3. Grouping of testcases is available in TestNG where as the same is not available in Junit.
4. *TestNG* It use “dependOnMethods “ to implement the dependency testing as following
5. Parallel provide by Tesng

@Test

(dependsOnMethods={"method1"})

**publicvoid** method2(){

System.out.println("This is method 2");

}

# [Difference between webdriver.Dispose(), .Close() and .Quit()](http://stackoverflow.com/questions/15067107/difference-between-webdriver-dispose-close-and-quit)

**1.** webDriver.Close() - Close the browser window that the driver has focus of  
**2 WebDriver.quit() method quits the driver, and closing every associated window.** (Close all the windows).  
**3.** webDriver.Dispose() Closes all browser windows and safely ends the session

In summary ensure that Quit() or Dispose() is called before exiting the program,

* **A bug is the result of a coding error**
* **A defect is a deviation from the requirements**

That is: A **defect does not necessarily mean there is a bug in the code**, it could be a function that was not implemented but defined in the requirements of the software. Not all software defects are caused by coding errors. One common source of expensive defects is caused by requirement gaps, e.g., unrecognized requirements, that result in errors of omission by the program designer

Defect-DeviationfromRequirements

**Difference between a defect and a failure**

When a defect reaches the end customer it is called a failure and if the defect is detected internally and resolved it’s called a defect.

**DOM vs HTML**

DOM is a model of a document with an associated API for manipulating it.

HTML is a markup language that lets you represent a certain kind of DOM in text.

Other kinds of DOMs can be expressed in other markup languages, for example RSS and Atom can be converted to a DOM and manipulated with the same API as an HTML or XHTML document (more or less anyway; there are some HTML specific DOM extensions).

The Document Object Model (DOM) is a language-independent model made up of objects representing the structure of a document. HTML is one language for writing such documents.

**DOM vs Xpath**

[DOM](http://en.wikipedia.org/wiki/Document_Object_Model) is the tree model to represent [HTML](http://en.wikipedia.org/wiki/HTML).

HTML is a markup language that lets you represent a certain kind of DOM in text.

XPath is simply the ability to traverse nodes from XMLAnd obtain information. Its whole base is for information retrieval.

DOM, whether XML or HTML, is based upon traversing for the purpose of reading or writing information.

**XPath** is not a query, but a query language for selecting nodes from an XML document.

DOM is just one tree model for XML,

XPath and DOM are both specifications, not implementations.

The **Document Object Model** (**DOM**) is *convention* for representing and interacting with [objects](http://en.wikipedia.org/wiki/Object_(computer_science)) in [HTML](http://en.wikipedia.org/wiki/HTML),[XHTML](http://en.wikipedia.org/wiki/XHTML) and [XML](http://en.wikipedia.org/wiki/XML) documents.

A document object model (DOM) is an application programming interface (API) for representing a document (such as an HTML document) and accessing and manipulating the various elements (such as HTML tags and strings of text) that make up that document. JavaScript-enabled web browsers have always defined a document object model; a web-browser DOM may specify, for example, that the forms in an HTML document are accessible through the forms[] array of the Document object;

**Fetch Data From excel**

Hi All,

<https://docs.google.com/document/d/1p9E_Ob9HevUVdOXOOylLMgqL7J_KTxygHj_f8PJr4lg/edit>

<http://executeautomation.com/blog/data-driven-testing-in-selenium-using-jxl-part-2/>

We have learned what is Data driven testing and how we can use JXL to read data from Excel Sheets in Part 1 of our previous post. But the greatest question of the day is, how to use JXL for data driven testing ?. How can I use this in my Selenium framework which I have? What are the steps which has to be taken care to make my framework a data driven framework?

Well, for this I don’t want to bore you guys with lot of theoretical contents which you can always find some way or other, but I would like to show the real working code.

In order to perform data driven testing, all we need to do is create a reusable library file just for Excel using JXL. The library file need to have following basic functionality in hand, let say

**Step 1 : Create a library file with all the below mentioned functionality**

1. Open Excel Sheet

2. Read Excel Sheet Row Count

3. Read Cell value from a specified location

4. Create a Dictionary to store Excel Sheet Column name

5. Create a function to read from the Dictionary

The Source code looks like this.

Next we are going to create actual test file which is going to perform intended operation, here we are going to perform Gmail login functionality.

**Step 2: Create a TestNG Class file to perform Gmail Login**

This TestNG class file should include

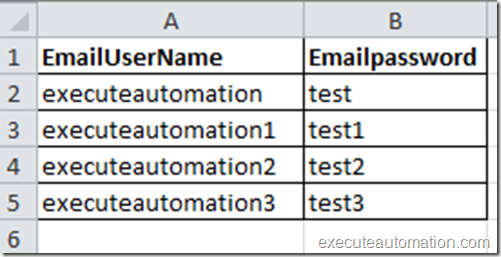
1. Opening a browser with Gmail

2. Perform User Name and password entry with different combinations of value by reading from Excel sheet

Source Code looks like this

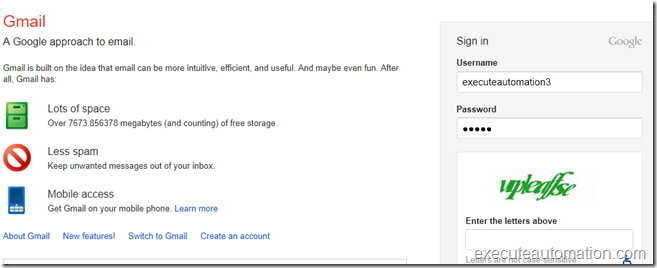
**package**DataDriver;  
  
  
**import**java.io.IOException;  
**import**jxl.read.biff.BiffException;  
  
**import**org.openqa.selenium.By;  
**import**org.openqa.selenium.WebDriver;  
**import**org.openqa.selenium.WebElement;  
**import**org.openqa.selenium.ie.InternetExplorerDriver;  
**import**org.testng.annotations.BeforeTest;  
**import**org.testng.annotations.Test;  
  
**publicclass** ReadDataTest {  
  
 *//Global initialization of Variables*  
 **static** ExcelSheetDriver xlsUtil;  
 WebDriver driver =**new** InternetExplorerDriver();  
  
 *//Constructor to initialze Excel for Data source*  
 **public** ReadDataTest()**throws** BiffException, IOException  
 {  
 *//Let's assume we have only one Excel File which holds all Testcases. weird :) Just for Demo !!!*  
    xlsUtil=**new** ExcelSheetDriver("D:**\\**Data.xls");  
    *//Load the Excel Sheet Col in to Dictionary for Further use in our Test cases.*  
    xlsUtil.ColumnDictionary();  
 }  
  
 @BeforeTest  
 **publicvoid** EnvironmentalSetup()  
 {  
  driver.get("http://www.gmail.com");  
 }  
  
 @Test  
  
 **publicvoid** GmailLoginPage()**throws**InterruptedException{  
  
  *//Create a for loop.. for iterate through our Excel sheet for all the test cases.*  
  **for**(**int** rowCnt =1;rowCnt &amp;lt; xlsUtil.RowCount();rowCnt++)  
  {  
  
   *//Enter User Name by reading data from Excel*  
   WebElement userName = driver.findElement(By.name("Email"));  
   userName.clear();  
   userName.sendKeys(xlsUtil.ReadCell(xlsUtil.GetCell("EmailUserName"), rowCnt));  
  
   *//Enter Password*  
   WebElement password = driver.findElement(By.name("Passwd"));  
   password.clear();  
   password.sendKeys(xlsUtil.ReadCell(xlsUtil.GetCell("Emailpassword"), rowCnt));  
  
   *//Click on the Sign In Button*  
   WebElement signin = driver.findElement(By.name("signIn"));  
   signin.click();  
  
   *//Sleep for some time,so that we can see things in action @ Screen :)*  
   Thread.sleep(2000);  
  }  
 }  
  
}

**Step 3: Create a actual Excel Sheet which holds the data to be supplied in TestNG class in above step**

Just create a Excel sheet, which looks something like this.

**Step 4 : Try executing the code**

Just try executing the code, Note in the above code I have place excel sheet in my D:\ drive, you can place it anywhere in machine and change the same in code.

Your Gmail screen looks like this at the end of test by executing all the above test data’s

Well you can download the full working source code from [here](http://executeautomation.com/downloads/DataDrivenTest.zip)

That’s it !!!

Happy coding.

# [Difference between @Before and @BeforeClass](http://stackoverflow.com/questions/20295578/difference-between-before-and-beforeclass)

The code marked @Before is executed before each test, while @BeforeClass runs once before the entire test fixture. If your test class has ten tests, @Before code will be executed ten times, but @BeforeClass will be executed only once. ©

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Group on classes  http://www.mkyong.com/unittest/testng-groups-example/  Group” can be applied on class level. In below example, every public method of this class “TestSelenium” is belong to group selenium-test. |  |  |
|  |  |  |

@Test(groups="selenium-test")

**Publicclass**TestSelenium{

**publicvoid** runSelenium(){

System.out.println("runSelenium()");

}

**Publicvoid**runSelenium1(){

System.out.println("runSelenium()1");

}

*<!-- Run test method on group "selenium" only -->*

**<test**name="selenium"**>**

**<groups><run><include**name="selenium-test"**/></run></groups>**

**<classes><class**name="com.mkyong.testng.examples.group.TestSelenium"**/><class**name="com.mkyong.testng.examples.group.TestGroup"**/></classes></test>**

}

**- Test methods**

Test methods are annotated with @Test. Methods annotated with @Test that happen to return a value will be ignored, unless you set allow-return-values to true in your testng.xml:

3[view source](http://testng.org/doc/documentation-main.html#viewSource)

[print](http://testng.org/doc/documentation-main.html#printSource)[?](http://testng.org/doc/documentation-main.html#about)

|  |  |
| --- | --- |
| <suite allow-return-values="true"> | |
|  |

|  |  |
| --- | --- |
| or | |
|  |

|  |
| --- |
| <test allow-return-values="true"> |

<http://saucelabs.com/resources/selenium/css-selectors>

<http://autotestgroup.com/en/blog/75.html>

**Advantages of POM**

**1.**Clear separation between test code and navigation code in code base.

**2.**Readability

3 Ease of maintenance, when UI changes you need to change the navigation code and not test (provided application

logic is still the same and only UI object/navigation is changed)

maintnance easy

4.eliminated duplication,Stay DRY. Page object model believes in the principle of Do not repeat yourself.

**Disadvantage of POM**:

**Takes time to build the infrastructure (as mentioned before)**

The only one I could think of is, initial design of page object might take more time but pain is definitely worth it

**What is the Document Object Model?**

The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM specification, the term "document" is used in the broad sense - increasingly, XML is being used as a way of representing many different kinds of information that may be stored in diverse systems,In the DOM, documents have a logical structure which is very much like a tree

**ThreadLocal :** The **ThreadLocal** class in Java enables you to create variables that can only be read and written by the same thread. Thus, even if two threads are executing the same code, and the code has a reference to a **ThreadLocal** variable, then the two threads cannot see each other's **ThreadLocal** variables.

**RemoteWebDriver driver = null;**

**ThreadLocal<RemoteWebDriver> driverThreadLocal = new ThreadLocal<RemoteWebDriver>();**

**driverThreadLocal.set(driver);**

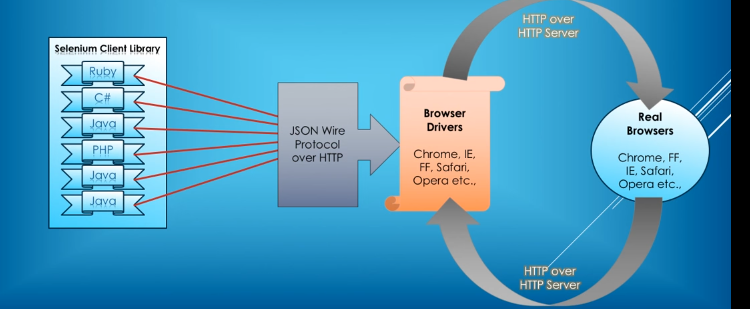
**public webdriver getdriver()**

**{**

**return driverThreadLocal.get();}**

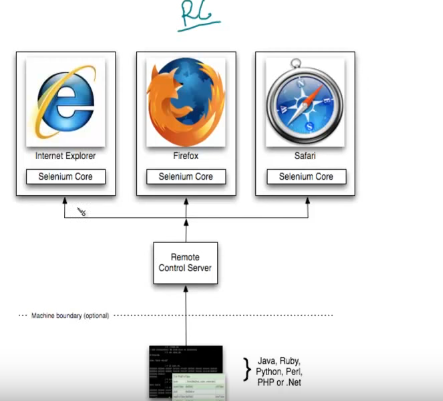
**}**

**How WebDriver works**

****

All the commands given by code(client library) converted with the help **JSON wire protocol** over HTTP converted to a URL and send to browser driver, these driver have internal HTTP server. for each command it internally pass to real browser.k

**How Selenium RC works**



**how to skip test cases in test ng**

2 ways we can do

1. by *@Test(enabled=false)*

2. [*Skip Exception*](http://testng.org/javadoc/org/testng/SkipException.html)***CONDITIONAL Skip***

@Test

**public** **void** **testCaseConditionalSkipException**()

{

int DataAvailable=7;

System.out.println("Im in Conditional Skip");

**if**(DataAvailable>5) **throw** **new** SkipException("Skipping this exception");

System.out.println("Executed Successfully"); }

 DataProvider

@DataProvider

public Object[][] getData()

{

Object[][] data = new Object[2][3];

data[0][0]=1;

data[0][1]=2;

data[0][2]=3;

data[1][0]=4;

data[1][1]=5;

data[1][2]=6;

//data[0][2]=3;

//data[0][3]=3;

return data;

}

**XSSF vs HSSF**

we use HSSF when we have to use file of format  
“.xls”  
but xssf is used for file formats  ” .xlsx

String excelFilePath = "Books.xlsx";

        FileInputStream inputStream = new FileInputStream(new File(excelFilePath));

        Workbook workbook = new XSSFWorkbook(inputStream);

        Sheet firstSheet = workbook.getSheetAt(0);

        Iterator<Row> rowiterator = firstSheet.iterator();

        while (rowiterator.hasNext()) {

            Row nextRow = rowiterator.next();

            Iterator<Cell> cellIterator = nextRow.cellIterator();

            while (cellIterator.hasNext()) {

                Cell cell = cellIterator.next();

                switch (cell.getCellType()) {

                    case Cell.CELL\_TYPE\_STRING:

                        System.out.print(cell.getStringCellValue());

                        break;

                    case Cell.CELL\_TYPE\_BOOLEAN:

                        System.out.print(cell.getBooleanCellValue());

                        break;

                    case Cell.CELL\_TYPE\_NUMERIC:

                        System.out.print(cell.getNumericCellValue());

                        break;

                }

                System.out.print(" - ");

            }

            System.out.println();

        }

        workbook.close();

        inputStream.close();

    }

All the latest Verison of New Drivers

Why Setter and getter used

**Explicit Wait**

**WebDriverWait wait = new WebDriverWait(*driver*, 30);**

**WebElement element =wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*id*("te")));**

This waits up to 30 seconds before throwing a TimeoutException or if it finds the element will return it in 0 - 30 seconds. **WebDriverWait by default calls the ExpectedCondition every 500 milliseconds until it returns successfully**. A successful return is for ExpectedCondition type is Boolean return true or not null return value for all other ExpectedCondition types.

**Fluent Wait**

Wait wait = **new** FluentWait(*driver*).withTimeout(5, TimeUnit.***SECONDS***).pollingEvery(4, TimeUnit.***SECONDS***);

WebElement foo = wait.until(new Function<WebDriver, WebElement>() {

public WebElement apply(WebDriver driver) {

return driver.findElement(By.id("Element"));

}

});}

You can use StopWatch object of org.apache.commons.lang3.time package. The following is the complete code of Selenium WebDriver using Java:

import org.apache.commons.lang3.time.StopWatch;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

public class TimerInSeleniumWebDriver {

public static void main(String[] args) {

WebDriver driver;

driver = new FirefoxDriver();

StopWatch pageLoad = new StopWatch();

pageLoad.start();

//Open your web app (In my case, I opened facebook)

driver.get("https://www.facebook.com/");

// Wait for the required any element (I am waiting for Login button in fb)

WebDriverWait wait = new WebDriverWait(driver, 10);

wait.until(ExpectedConditions.presenceOfElementLocated(By.id("u\_0\_l")));

pageLoad.stop();

//Get the time

long pageLoadTime\_ms = pageLoad.getTime();

long pageLoadTime\_Seconds = pageLoadTime\_ms / 1000;

System.out.println("Total Page Load Time: " + pageLoadTime\_ms + " milliseconds");

System.out.println("Total Page Load Time: " + pageLoadTime\_Seconds + " seconds");

driver.close();

}

}

**IF need to run url for 2 seconds only**

driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.MILLISECONDS);

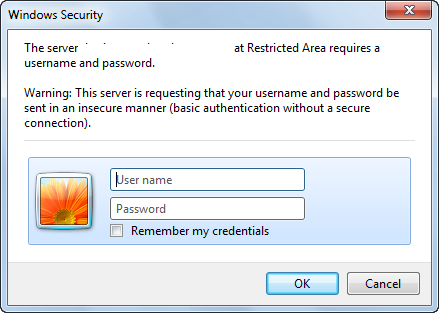
# Handling Authentication Window with WebDriver

To work with Basic Authentication pop-up (which is a browser dialogue window), you just need to send the user name and password along with the application URL. for chrome and Firefox

driver.**get**("[http://admin:admin@yoururl.com"](http://admin:admin@yoururl.com/));

IE

The best way to make it work with IE is using AutoIt tool. If not, you may need to change the stuff in registry, To change for the current user, you need to edit in 'HKEY\_CURRENT\_USER...' and if you want to do that for all users, you can set the value of register keys as 'HKEY\_LOCAL\_MACHINE...' etc.

Once you open the URL in IE it will look like the below screen shot: -  


Check out the example to work with IE using AutoIt tool.

First create AutoIt script as below and save it as basicauth.au3

; To **pass** user name **and** password

WinWaitActive("Windows Security")

Send("admin")

Send("{TAB}")

Send("admin")

Send("{ENTER}")

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

element.click();

               //Which calls the autoit exe file

Runtime.getRuntime().exec("G:/Tutorial/AutoItScripts/upload.exe");

}

}

The below is the AutoIt script:

WinWaitActive("File Upload")

Send("G:\Tutorial\AutoItScripts\TestScripts\Test.doc")

Send("{ENTER}")

Page Object Model : a Design pattern : segregate selenium code based on pages.

Page Factory: Advanced concept ( POM + new features )

1. Elements are identified using @FindBy Annotation

2. Initialism all the elements declared in Point#1 at a time.

( in POM, initialization happens on the fly )

PageFactory.initElements(driver,this);

Thats all.

**What Is StaleElementReferenceException?**

Stale means old or we can say no longer fresh element. Let me describe you In very simple words. Example : You have a search text box on software web page. When you search for some keyword, text box's position get changed on page. So In this case, Look and feel, Identifiers etc. of text box will remain same but what Internally happened Is -> JS library has deleted previous text box and replaced It with new same text box. So now If you will go to use same text box using previously located reference In your software test, You will get **StaleElementReferenceException**In console.

Use explicit wait to getaway From this

**A WebElement is a reference to an element in the DOM.**

**A StaleElementException is thrown when the element you were interacting is destroyed and then recreated. Most complex web pages these days will move things about on the fly as the user interacts with it and this requires elements in the DOM to be destroyed and recreated.**

**When this happens the reference to the element in the DOM that you previously had becomes stale and you are no longer able to use this reference to interact with the element in the DOM. When this happens you will need to refresh your reference, or in real world terms find the element again.**

If you will run bellow given example, It will throw **StaleElementReferenceException**.

package Testing\_Pack;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class StaleElement {

WebDriver driver;

@BeforeTest

public void setup() throws Exception {

driver =new FirefoxDriver();

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

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

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

}

@Test

public void getExe() throws InterruptedException{

**//Located element and stored It's reference In variable.**

WebElement Search\_Box = driver.findElement(By.xpath("//input[@name='q']"));

**//Used element reference variable to locate element and perform search.**

Search\_Box.sendKeys("Hello");

Search\_Box.sendKeys(Keys.ENTER);

Thread.sleep(5000);

**//After search operation, Element's position Is changed.**

**//Now I am using same reference variable to clear search text box.**

**//So here, WebDriver will be not able to locate element using same reference and It will throw StaleElementReferenceException.**

Search\_Box.clear();

}

}

**Ways to Overcome StalElementException**

Kenny's solution is good, however it can be written in a more elegant way

new WebDriverWait(driver, timeout)

.ignoring(StaleElementReferenceException.class)

.until((WebDriver d) -> {

d.findElement(By.id("checkoutLink")).click();

return true;

});

Or also:

new WebDriverWait(driver, timeout).ignoring(StaleElementReferenceException.class).until(ExpectedConditions.elementToBeClickable(By.id("checkoutLink")));

driver.findElement(By.id("checkoutLink")).click();

**Solution 1:**

You could refresh the page and try again for the same element.

**driver.navigate().refersh();**

**driver.findElement(By.xpath("xpath here")).click();**

**Solution 2:**

If an element is not attached to DOM then you could try using ‘try-catch block’ within ‘for loop’



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | // Using for loop, it tries for 3 times.  // If the element is located for the first time then it breaks from the for loop nad comeout of the loop  for(int i=0; i<=2;i++){    try{       driver.findElement(By.xpath("xpath here")).click();       break;    }    catch(Exception e){       Sysout(e.getMessage());    }  }  **3.** Wait for the element till it gets available  **wait.until(ExpectedConditions.presenceOfElementLocated(By.id("table")));** |

**Page Object Model** : a Design pattern : segregate selenium code based on pages.

Ex: Create a separate java class for Login page , one more class for Home Page etc.

**Page Factory:** Selenium Page Factory Pattern is like an extension to [Page Object Model](http://www.seleniumeasy.com/selenium-tutorials/page-object-model-framework-introduction) , but Page Factory is much enhanced model. To start with, we just need to import package ‘org.openqa.selenium.support.PageFactory’

# Advanced concept ( POM + new features )== Page Factory Design Pattern (Enhanced POM)

1. Elements are identified using @FindBy Annotation

2. Initialism all the elements declared in Point#1 at a time.

**( in POM, initialization happens on the fly )**

**PageFactory.initElements(driver,this);**

Or,

// To initialize elements.

HomePage homePage = PageFactory.initElements(driver, HomePage.class);

**Or**, **as a constructor for page class as below:**

public HompePage(WebDriver driver) {

this.driver = driver;

PageFactory.initElements(driver, this);

}

**What is Keyword Driven Framework?**

Keyword Driven Framework is a type of **Functional Automation Testing Framework** which is also known as **Table-Driven** testing or **Action Word based** testing. The basic working of the *Keyword Driven Framework* is to divide the Test Case in to four different parts. First is called as *Test Step*, second is *Object* of Test Step, third is *Action* on Test Object and fourth is *Data* for Test Object.

**The above categorization can be done and maintained with the help of Excel spread sheet:**

**Test Step:** It is a very small description of the *Test Step* or the description of the *Action* going to perform on *Test Object*.  
**Test Object:** It is the name of the Web *Page object/element*, like Username & Password.  
**Action:** It is the name of the *action*, which is going to perform on any Object such as *click*, *open* browser, *input*etc.  
**Test Data:** Data can be any value which is needed by the Object to perform any action, like Username value for Username field.

**Exceptions**

**Different Exceptions in Selenium**

There is a complete list of *Exceptions* mentioned on the Selenium Doc which you may or may not encounter in course of your  testing.

**Most common Exceptions:**

1. ***NoSuchElementException****:* FindBy method can’t find the element.
2. ***StaleElementReferenceException****:* This tells that element is no longer appearing on the DOM page.
3. ***TimeoutException****:* This tells that the execution is failed because the command did not complete in enough time.
4. ***ElementNotVisibleException****:*Thrown to indicate that although an element is present on the DOM, it is not visible, and so is not able to be interacted with
5. ***ElementNotSelectableException****:* Thrown to indicate that may be the element is disabled, and so is not able to select.
6. **ElementNotSelectableException**(*msg=None*, *screen=None*, *stacktrace=None*)[[source]](https://seleniumhq.github.io/selenium/docs/api/py/_modules/selenium/common/exceptions.html#ElementNotSelectableException) Thrown when trying to select an unselectable element.
7. **NoSuchAttributeException--** **Thrown when the attribute of element could not be found.**
8. **NoSuchFrameException**
9. **NoSuchWindowException**

**How to Handle Exception**

**1.Try/Catch**

**2.Multiple catch block**

**3.** **Throw:**Sometimes we want to generate exception explicitly in our code, for example in Selenium Automation Framework most of the time we print self-written logs, once we catch an exception and then we need to throw that exception back to the system so that the test case can be terminated. Throw keyword is used to throw exception to the run time to handle it.

**4.Throws:**When we are throwing any exception in a method and not handling it, then we need to use throws keyword in method signature to let caller program know the exceptions that might be thrown by the method.

public static void openBrowser(String object){

try{

Log.info("Opening Browser");

driver=new FirefoxDriver();

//This block will execute only in case of an exception

}catch(Exception e){

//This is to print the logs - Method Name & Error description/stack

Log.info("Not able to open Browser --- " + e.getMessage());

//Set the value of result variable to false

**DriverScript.bResult = false;**

**}**

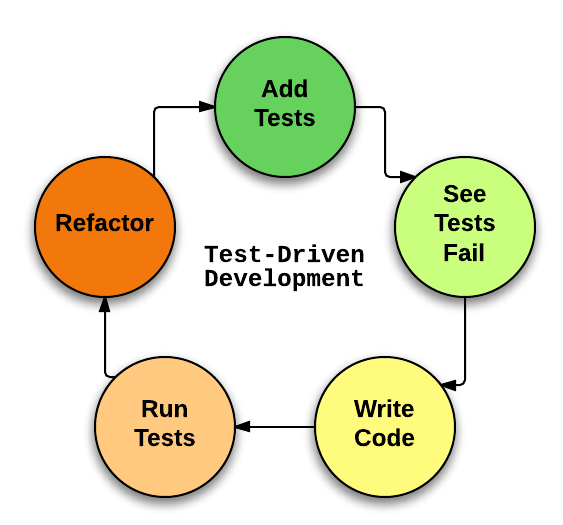
}

**Test-Driven Development**

When I first heard about TDD, the idea seemed to be pretty simple. Just by doing a little word swizzling, obviously TDD is when you have tests that drive your software development.

If we were to unpack the definition of TDD a bit more, we’d see that it is usually broken up into five different stages:

1. First the developer writes some tests.
2. The developer then runs those tests and (obviously) they fail because none of those features are actually implemented.
3. Next the developer actually implements those tests in code.
4. If the developer writes their code well, then the in next stage they will see that their tests pass.
5. The developer can then refactor their code, add comments and clean it up, as they wish because the developer knows that if the new code breaks something, then the tests will be an alert by failing.



**Behavior-Driven Development--**

[Cucumber](https://en.wikipedia.org/wiki/Cucumber_%28software%29) (Ruby framework) ;[JBehave](https://en.wikipedia.org/w/index.php?title=JBehave_%28software%29&action=edit&redlink=1),[[1]](https://en.wikipedia.org/wiki/Behavior-driven_development#cite_note-BDD_Def_BehaviourDriven-1) ; RSpec

 [Behat](https://en.wikipedia.org/wiki/Behat_%28computer_science%29) (PHP framework)

 [Jasmine (JavaScript testing framework)](https://en.wikipedia.org/wiki/Jasmine_%28JavaScript_testing_framework%29)

 [Concordion](https://en.wikipedia.org/wiki/Concordion) (Java framework)

 [Squish GUI Tester](https://en.wikipedia.org/wiki/Squish_%28Froglogic%29)

Alright, so what is BDD you ask? Well that’s where the line gets a little fuzzy. Some people will say it is similar to TDD, others will say that it **is** just TDD but with better guidelines, or even a totally different approach to developing.

Whatever the actual definition is, it doesn’t matter that much. The main thing to know is that **BDD is meant to eliminate issues that TDD might cause.**

The main difference is just the wording. BDD uses a more verbose style so that it can be read almost like a sentence

Although this example is very simple and doesn’t illustrate it, BDD tests should be more focused on the features, not the actual results. Often you’ll hear that BDD is to help **design** the software, not test it like what TDD is meant to do.

**Unit testing**

A unit test focuses on a single “unit of code” – usually a function in an object or module. By making the test specific to a single function, the test should be simple, quick to write, and quick to run. A unit test should be isolated from dependencies – for example, no network access and no database access.

Mocha Framework

it world as TDD and BDD both

**Interfaces**

Mocha’s “interface” system allows developers to choose their style of DSL. Mocha has **BDD**, **TDD**, **Exports**, **QUnit** and **Require**-style interfaces.

**BDD**

The **BDD** interface provides describe(), context(), it(), specify(), before(), after(), beforeEach(), and afterEach().

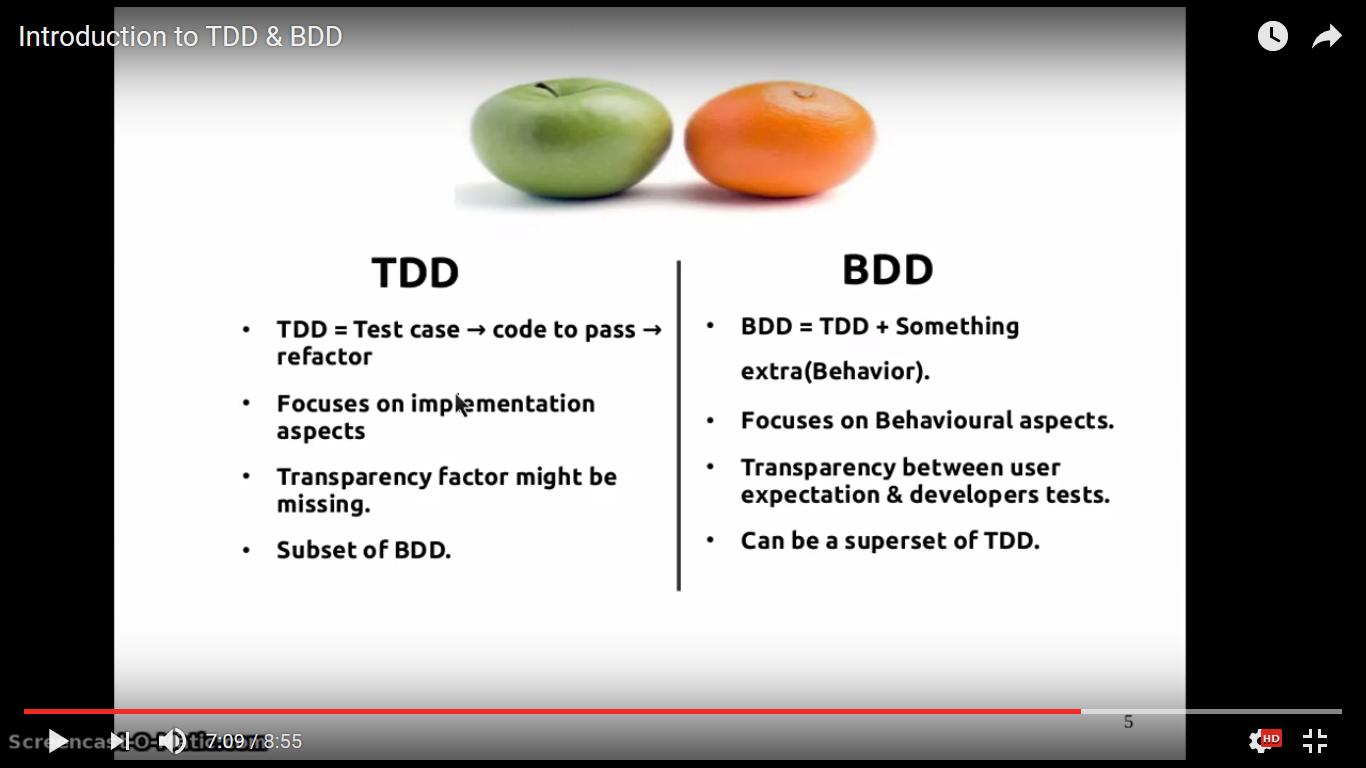
**TDD**

The **TDD** interface provides suite(), test(), suiteSetup(), suiteTeardown(), setup(), and teardown():

context() is just an alias for describe(), and behaves the same way; it just provides a way to keep tests easier to read and organized. Similarly, specify() is an alias for it().







# [Does a testng annotation can be inheried?](http://stackoverflow.com/questions/29234903/does-a-testng-annotation-can-be-inheried)

Java does not inherit Annotations without the [@Inherited](http://docs.oracle.com/javase/7/docs/api/java/lang/annotation/Inherited.html) meta-annotation from interfaces, a pertinent question can be found [here](http://stackoverflow.com/questions/4745798/why-java-classes-do-not-inherit-annotations-from-implemented-interfaces)

So if you wanted to get inherited annotations you could write custom annotations that grant the functionality of the @beforeXXX or @afterXXX annotations that also have the @Inherited annotation

**What is Annotations ?**

**Annotations are *meta-meta-objects* which can be used to describe other *meta-objects*. Meta-objects are classes, fields and methods. Asking an object for its meta-object (e.g. anObj.getClass() ) is called *introspection*. The introspection can go further and we can ask a meta-object what are its annotations (e.g. aClass.getAnnotations). Introspection and annotations belong to what is called *reflection* and *meta-programming*.**

The interface keyword indicates that you are declaring a traditional interface class in Java.

**The @interface keyword is used to declare a new annotation type.**

**What's the difference between interface and @interface in java?**

The *@* symbol denotes an annotation type definition.

That means it is **not** really an interface, but rather a new annotation type -- to be used as a function modifier, such as *@override*.

**how to handle dynamic element in selenium webdriver?**

1. Absolute XPath

2. Identify Element by starting Text

3.<input class="new-userfield-001">

<input class="old-userfield-002">

As we can see ‘usefield’ part of element is static, so we can apply ‘contains’ function to access this element locator as shown below…

XPath: //input[contains(@class, '-userfield-')].

4. Identify Element by Index

driver.findElements(By.xpath(“//\*submit”)).get(0).click();

5.We can use the DOM structure to find the closest stable element first and then this stable element can be used as a reference element to find the required element.

XPATH: //span1/../following-sibling::div//button1

**Optimzation of Framework**

1. Extra wait/Sleep time reduce

2. Change locator type from xpath to id .

3) Focus your Optimization Effort---Another way to look at the test performance issue could be to start by identifying the tests or the functions/methods/keywords that are the more time consuming over your whole portfolio. Focussing your effort on those parts could lead to quick wins. Here are two

4) All data in buffer from excel

5)Shared Setup and Teardown

6) Avoid Sleep

7) 3rd party tool check if taking more time

**Database upgradations:**

1) Test the upgrade itself.  Do problems arise?  Has anything new been introduced that needs to be accounted for?  If so, make sure it works properly.   
2) Run functional tests that rely on the database to ensure they function properly.     
3) Run performance tests to ensure data is obtained within specified performance parameters.    
4) Ensure data integrity is maintained after upgrade.  Contstraints, for example, should still be handled as expected.  
  
5) Run load and/or stress test to ensure the upgraded database still handles user load and performs well under the most business-critical resource-intensive circumstances.   
6) Test the Oracle client.  Ideally, any issues will have been caught in the upgrade test, but it is a necessary step in the process.

@Test(threadPoolSize = 3, invocationCount = 10, timeOut = 10000)

1. **the method will be run a total of 10 times using 3 threads**

**REports**

**How to customize TestNG Report**

TestNG reporting is quite handy but still, sometimes we need some less data in reports or want to display reports in some other format like pdf, excel, etc. or want to change report's layout

**There can be two ways we can customize TestNG report**

* **Using ITestListener Interface:**
* **Using IReporter Interface:**

**ITestListener Interface**

* We use this interface when we need to customize real time report. In other words, if we are executing the bunch of test cases in a TetNG suite and we want to get the report of each test case, then after each test case we need to implement ITestListener interface. This interface will override onTestFailure, onTestStart
* , onTestSkipped method to send the correct status of the current test case.
* **IReporter Interface**
* If we want to customize final test report generated by TestNG, we need to implement IReporter interface. This interface has only one method to implement generateReport. This method has all the information of a complete test execution in the List<ISuite>, and we can generate the report using it.

**IREporter vs ITestListener**

IReporter works \_after\_ your suite but iTestListener \_during\_   
  
In other words,  IReporter can just draw your report based on the   
results in ISuite but ITestListener can catch and store more info, say   
screenshots taken on test failure.

**Interface ‘ITestListener’ vs extended**[**‘TestListenerAdapter’**](http://testng.org/javadocs/org/testng/TestListenerAdapter.html)

You can either extend ['TestListenerAdapter'](http://testng.org/javadocs/org/testng/TestListenerAdapter.html)or implement Interface ['ITestListener'](http://testng.org/javadocs/org/testng/ITestListener.html) which is a listener for test running.

We have extended TestListenerAdapter which intern implements ITestListener with empty methods. So again we don't have to override other methods from the ITestListener interface which we may not needed.

selenium-server-standalone.jar

Why Selenium-Server compared to Selenium-WebDriver?

It is mainly use in the Selenium Grid. As we use different OS and browsers with different machines, we need to run it same time for various reason. In selenium Grid, we use the machine Hub and Node(You can go through selenium grid docs for more), so for running the selenium in different machines with the main machine, we need server standalone.

Reasons are:

1. You are using Selenium-Grid to distribute your tests over multiple machines or virtual machines (VMs).
2. You want to connect to a remote machine that has a particular browser version that is not on your current machine.
3. You are not using the Java bindings (i.e. Python, C#, or Ruby) and would like to use HtmlUnit Driver.

# what is ZAPI ? is ZAPI same as ZBot ?

ZAPI is an add-on to Zephyr for JIRA Server, that allows access to its testing data programmatically via RESTful APIs. Please access the below link to know more about ZAPI.

**The Concept**

Zephyr is based around a concept of Desktops & Dashboards. Every role in a Test/QA Department has a customized web-based Desktop with relevant applications that allow them to do their jobs quickly and efficiently, as they all share data from a centralized Zephyr Server and communicate via a collaborative backbone. Dashboards are automated and live, keeping the whole company updated on every aspect of testing and product quality. The Zephyr Server offers integrations to a JIRA Defect Tracking systems. The ZBot, ZIP technologies, and documented APIs allows integration with various automation tools.

**Roles**

By default, the users of Zephyr fall typically in 5 roles: Test Manager, Test Lead, Tester, Defect and Dashboard user. Custom roles can be defined as needed in the system. Test Managers set up their test resources in the system and assign them to various projects that are being undertaken by the Test Department. Emails are automatically sent to them with information on how to log into their Desktops. Test Leads and Testers now log into Zephyr and are presented with project/release areas with relevant applications to help them effectively and collaboratively do their jobs. Defect users use the Defect Tracking System in their Desktops. Dashboard users are only allowed access to Dashboards. Roles for Developers, Project Managers and Business personnel can also be created and allowed access to specific applications thereby allowing them to interact with data.

https://zephyrdocs.atlassian.net/wiki/spaces/ZE61/pages/263522555/Zephyr+Vortex

# [How to get the test result status from TestNG/Selenium in @AfterMethod?](https://stackoverflow.com/questions/36125229/how-to-get-the-test-result-status-from-testng-selenium-in-aftermethod)

public class stacktest {

@Test

public void teststackquestion() {

boolean actual = true;

boolean expected = false;

Assert.assertEquals(actual, expected);

}

@AfterMethod

public void afterMethod(ITestResult result)

{

try

{

if(result.getStatus() == ITestResult.SUCCESS)

{

//Do something here

System.out.println("passed \*\*\*\*\*\*\*\*\*\*");

}

else if(result.getStatus() == ITestResult.FAILURE)

{

//Do something here

System.out.println("Failed \*\*\*\*\*\*\*\*\*\*\*");

}

else if(result.getStatus() == ITestResult.SKIP ){

System.out.println("Skiped\*\*\*\*\*\*\*\*\*\*\*");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}