### How to explain Selenium Automation Framework (DataDriven+POM) in Interview

<https://www.automationtestinginsider.com/2020/09/how-to-explain-selenium-automation.html>

<https://techcanvass.com/blogs/Automation-testing-resource-library.aspx>

<https://www.youtube.com/channel/UCrBErf8YupvzR9UcMxIkDjg>

**Important points and tips to explain selenium automation Framework in Interviews**

<https://www.youtube.com/watch?v=xSN3wzj9Kn8>

1. Programming Language

Classes

Inheritance: TestBase.java

Access Modifiers (public, private and protected)

Constructors (Parameterized)

Loggers

Reports

Property Files

Read File

1. Automation Framework

Modular Driven: Reusable Functions

Data Driven: Maintain data in excel file

Hybrid

1. Design Pattern

Page Object Model with Page Factory

1. TDD/BDD [TetNG / Cucumber]

@BeforeSuite, @BeforeTest, @BeforeClass, @BeforeMethod, @Test, @Dataprovider

1. Maven Project

pom.xml

dependencies

plugins

profiles

Directory Structure

src/main/java: All the java files

src/main/resources: property files, xml fles

src/test/java: All the Test Cases

1. Reporting

Extent Report

1. CI/CD

Maven and Jenkins

Organization: Hexaware

Module 1: User Creation, User Mapping, Template creation and mapping

Module2: Report scheduling.

Module3: Verify the UC4 Job status.

Module4: Verify the export as csv/pdf functionality for scheduled reports in the respective TRUI.

Folder Structure

src/main/java

BaseClass, Config, Constants, Pages, Reports, RetryLogic, TestData, TestListeners, Utilities

Src/test/java

TestCases

src/main/resources

log4j.properties, testng.xml, testing\_sanity.xml

src/main/java

com.hexa.grpqa.Base

TestBase.java

TestBase() constructor contains reading the property file

Initialization() contains the following

Instantiating the browser, Register the driver with EventFiringWebDriver, Maximize window, deleteAllCookies, pageLoadTimeOut, implicitlyWait, get().

com.hexa.grpqa.Config

Browser Name, Url, UserName & Password

com.hexa.grpqa.Constants

driverPath, Page Load TimeOut, Waits, Page Titles, Test Data Sheet Path

com.hexa.grpqa.pages

NewUserPage.java

NewUserMappingPage.java

NewTemplatePage.java

NewTemplateMappingPage.java

ReportsOptinPage.java

com.hexa.grpqa.Reports

ExtentReport Setup

com.hexa.grpqa.RetryLogic

MyTransformer.java implements IAnnotionTransformer and has transform() method. Inside the method call setRetryAnalyzer () method.

RetryAnalyzer.java implements IRetryAnalyzer and has retry() method.

com.hexa.grpqa.TestData

GRPTestData.xlsx

com. Hexa.grpqa.TestListeners

ExtentReportListeners.java extends ExtenReportSetup implements ITestListeners

onTestStart(ITestResult result), onTestSuccess(ITestResult result), onTestFailure(ITestResult result), onTestSkipped(ITestResult result),

onTestFailedButWithinSuccessPercentage(ITestResult result), onStart(ITestContext context), onFinish(ITestContext context)

com.hexa.grpqa.Utilities

JavaScriptUtilities.java

RandomUtilities.java

StaticWaits.java

TestUtility.java

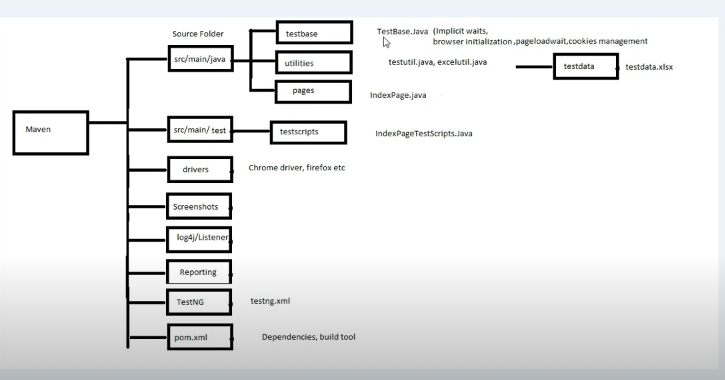
getTestData(), switchToFrame(), takeScreenshotAtEndOfTest(), clickOn(), sendKeys(), waitForElementsToBeVisible(), switchWindow(), getScreenshot(),

getSystmDate(), setDateForLog4j().

WebEventListeners.java extends TestBase implements WebDriverEventListener

XlsReader.java

getRowCount(), getCellData(), setCellData(), addSheet(), removeSheet(), addColumn(), removeColumn(), isSheetExists(), getColumnCount(), getCellRowNo()



**To build a Framework do the following**

1. Create Maven Project
2. Create Page object files
3. Implement Reusable Utilities and Centralized variables
4. Write Test Cases
5. Convert into Test NG Framework
6. Implement Data driven / Parameterizing Practices
7. Inject Log4J logging
8. Use Test NG Listeners and capture Screenshots on Test Failures
9. Generate Reports using Extent Report, Report NG for Test Results
10. Integrate with Jenkins

**Common Challenges in Selenium Automation**

1. Sync Issue or Timeout
2. Identifying Dynamic Elements

The dynamic elements can be handled with dynamic XPath or dynamic CSS selectors. You can also use functions, such as contains, ends with, starts-with, etc. to handle dynamic objects.

1. Cross Browser Testing

Because few locators will work in one browser but not on the other. Nowadays we have SauceLabs, BrowserStack which provides as complete infrastructure like Windows, MAC, Linux with different browser combinations as well.

1. DOM Structure/Locators

I collaborated with developers and came up with adding New attribute for elements for automataion purpose. E.g. TestID or ID

1. Framework Enhancement & Maintenance

Framework designing and maintenance is not a one-day activity, we have to keep on adding new features or libraries so that we can minimize execution time and maintenance task.

1. Popup Windows
2. Page Loading
3. Scalability

The primary goal of automation testing is to execute as many tests as possible for extensive test coverage in minimum time. With Selenium Web Driver, you can sequentially perform automation testing. So, Selenium is not the most scalable approach for automation testing.

1. Multi Tab Testing
2. Limited Reporting

**TestNG Tutorials**

**What is TestNG?**

* TestNG is a testing framework designed to simplify a range of testing needs from unit testing to integration testing.
* TestNG is a testing framework that can be integrated with selenium or any other automation tool to provide multiple capabilities like assertions, reporting, parallel test execution, etc.

**Advantage / Feature of TestNG**

1. Annotations are easier to use and understand.
2. Test Cases can be grouped more easily. e.g. Group=Sanity (20 TCs) & Group=Regression (100 TCs)
3. Provides a feature to create and execute parallel tests.
4. Supports Parameterization of Testcases.
5. Generates very good html reports which are useful to showcase to the stake holders.
6. Allows defining dependency of one test method over other methods.
7. Supports assigning priority to test cases.
8. Testng executes the @Test methods in alphabetical order.
9. Once execution is done select the project and Press f5(Refresh) to the see the test-output folder.
   1. index.html
   2. emailable-report.html

**What is the use of testng.xml file?**

Testng.xml is used for configuring the whole test suite. The various uses of TestNG are as follows

* All the tests in the test suite are triggered by testing.xml
* Used to pass parameters to test scripts.
* Supports the inclusion and exclusion of tests.
* Used to create test groups.
* Supports the parallel execution of test cases.

**What are Annotations in TestNG?**

Annotations in TestNG are a set of code that controls how method below them have to be executed i.e, the order of execution of methods below them will be decided by annotations that we give.

**What is the sequence of execution of annotations in TestNG?**

@BeforeSuite

@BeforeTest

@BeforeClass

@BeforeMethod

@Test

@AfterMethod

@AfterClass

@AfterTest

@AfterSuite

**How can we pass parameter to test script using TestNG?**

We can pass the parameter to test scripts by using @Parameter annotation in test and “parameter” tag in testing.xml.

Sample testing.xml

<suite name = “dlaTestSuite”>

<test name = “dlaTest”>

<parameter name = “dlaParamName” value = “dlaParamValue”>

<classes>

<class name = “dlaTestFile” />

</classes> </test> </suite>

Sample Test Script

public class dlaTestFile{

@Test

@Parameters(“dlaParamName”)

public void dlaParameterTest(string paramValue){

System.out.println(sampleParamName);

**What is difference between Suite, Test, and Class?**

Suite: A suite is made of one or more tests.

Test: A test is made of one or more classes.

Class: A class is made of one or more methods

**How can we create data driven framework using TestNG?**

To create a data-driven framework, @DataProvider is used in which data is passed to the associated test method and multiple iterations of tests run for different data values passed from @DataProvider method.

@DataProvider(name = “dlaDataProvider”)

public Object[] [] dataProviderMethod() {

return new Object[] [] {{“dev”, “lab”}, {“devlabs”, “alliance”}};

}

@Test(dataProvider = “dlaDataProvider”)

public void dlaTest(string s1, string s2) {

system.out.println(s1 + “ “ + s2);

**What is the difference between @BeforeClass and @BeforeMethod?**

1. The method with @BeforeClass will be executed only once before any of the tests in the current class are run whereas, a method annotated with @BeforeMethod will be executed before each method annotated with @Test.

2. @BeforeClass annotation can be used to set up the configuration and initialization which is common to all test methods in the current class. For example, we can set up driver configuration which will be common for all tests in the class.

@BeforeMethod can be used to set that data which is repeating before each @Test annotated method.

**What is the use of @Listener annotation in TestNG?**

1. Listeners are basically the ones who have the ability to listen to a particular event.
2. It is defined as an interface that modifies the behavior of the system.
3. Listeners allow customization of reports and logs.
4. Listeners mainly comprise of two types: WebDriver listeners & TestNG listeners

WebDriverEventListener interface allows to implement methods and classes like WebDriverEventListener and EventFiringWebDriver.

**WebDriverEventListener**

* This is an interface which holds some predefined methods. These Webdriver events are helpful to view the events triggered by the webdriver.
* It plays an important role in analyzing the results and helps us in debugging issues if we encounter any.
* It has ability to track different events like “beforeNavigateTo” , “afterNavigateTo”, “BeforeClickOn”, “AfterClickOn” and many more.

**TestNG Listeners**

TestNG can be made to listen to what we say with the help of Listeners. Listeners give us the flexibility to alter the default TestNG behavior.

* TestNG listener is formally called as ITestListener, which is an interface in TestNG.
* A normal Java class implements ITestListener and overrides all the corresponding methods written inside it.

Following are the scenarios that can be made

* If the test case is failed, then what action should be performed by the listener.
* If the test case is passed, then what action should be performed by the listener.
* If the test case is skipped, then what action should be performed by the listener.

import org.testng.Assert;

import org.testng.annotations.Listeners;

import org.testng.annotations.Test;

@Listeners(GrpListener.class)

public class TestCases

{

public WebDriver driver;

//Test to pass as to verify listeners.

@Test

public void Login() throws Exception

{

System.setProperty("webdriver.chrome.driver", "C:UsersVaishnaviDownloadschromedriver\_win32 (2)chromedriver.exe");

driver = new ChromeDriver();

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

JavascriptExecutor js = (JavascriptExecutor) driver;

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

driver.findElement(By.cssSelector("#search-inp")).sendKeys("Test Automation Engineer Masters Program");

driver.findElement(By.className("typeahead\_\_button")).click();

js.executeScript("window.scrollBy(0,500)");

Thread.sleep(3000);

js.executeScript("window.scrollBy(0,700)");

Thread.sleep(4000);

js.executeScript("window.scrollBy(0,700)");

Thread.sleep(4000);

}

//Forcefully failed this test as verify listener.

@Test

public void TestToFail()

{

System.out.println("This method to test fail");

Assert.assertTrue(false);

}

}

**Listener.java**

import org.testng.ITestContext;

import org.testng.ITestListener;

import org.testng.ITestResult;

public class GrpListener implements ITestListener

{

@Override

public void onTestStart(ITestResult result) {

System.out.println("onTest Start : "+result.getName());

}

@Override

public void onTestSuccess(ITestResult result) {

System.out.println("Success of test cases and its details are : "+result.getName());

}

@Override

public void onTestFailure(ITestResult result) {

System.out.println("Failure of test cases and its details are : "+result.getName());

}

@Override

public void onTestSkipped(ITestResult result) {

System.out.println("Skip of test cases and its details are : "+result.getName());

}

@Override

public void onTestFailedButWithinSuccessPercentage(ITestResult result) {

System.out.println("Failure of test cases and its details are : "+result.getName());

}

@Override

public void onStart(ITestContext context) {

}

@Override

public void onFinish(ITestContext context) {

}}

**Difference between WebDriver and TestNG listeners**

|  |  |
| --- | --- |
| **TestNG Listeners** | **WebDriver Listeners** |
| Used to generate the report for tests. Helps in capturing screenshots. | Performs jobs like TestNG listeners like logging in and reporting but works on different events |

**How can we run test cases in parallel using TestNG?**

To run the tests in parallel in TestNG, we have to add these two key-value pairs in the suite-

Parallel = “{methods/tests/classes}“

thread-count= “{number of thread you want to run simultaneously}“

<suite name = “DLATestSuite” parallel = “methods” thread-count = “5”>

**How can we make sure a test method runs even if the test methods or groups on which it depends fail or get skipped?**

To run the test method even if test methods or groups on which it depends get fail or skipped, we use “alwaysRun” attribute of @Test annotation.

@Test

public void parentTest() {

Assert.Fail(“Failed Test”);

}

@Test(dependsOnMethods = {“parentTest”}, alwaysRun = true) Public void DependentTest() {

System.out.println(“Test DLA”);

}

**How to handle exceptions in TestNG?**

To handle exception in methods we can mention the exception in @Test annotation so that the test case does not fail.

E.g. If a test method is expected to have “numberFormatException” exception, then the test case will fail because of this exception if no try-catch block is specified.

But this can be handled in TestNG by using “expectedException” attribute

@Test(expectedException=numberFormatException.class)

After this, the test case will run without failing.

**What is threadPoolSize? How can we use it?**

Ans. The threadPoolSize attribute specifies the number of threads to be assigned to the test method. This is used in conjunction with invocationCount attribute. The number of threads will get divided with the number of iterations of the test method specified in the invocationCount attribute.

@Test(threadPoolSize = 5, invocationCount = 10)

public void threadPoolTest(){

//Test logic

}

**How can we skip a test case conditionally?**

Ans. Using SkipException, we can conditionally skip a test case. On throwing the skipException, the test method is marked as skipped in the test execution report and any statement after throwing the exception will not get executed.

@Test

public void testMethod(){

if(conditionToCheckForSkippingTest)

throw new SkipException("Skipping the test");

//test logic

}

**What is the use of @Factory annotation in TestNG?**

* The @Factory annotation is useful when we want to run multiple test cases through a single test class.
* It is mainly used for the dynamic execution of test cases.

**testcase1.java**

import org.testng.annotations.Test;

public class Testcase1

{

@Test

public void test1()

{

System.out.println("testcase 1");

}

}

**testcase2.java**

import org.testng.annotations.Test;

public class Testcase2

{

@Test

public void test1()

{

System.out.println("testcase 2");

}

}

**Factory.java**

import org.testng.annotations.Factory;

public class Factory1

{

@Factory

public Object[] getTestClasses()

{

Object tests[]=new Object[2];

tests[0]=new Testcase1();

tests[1]=new Testcase2();

return tests;

}

}

**What is the difference between @Factory and @DataProvider annotation?**

@DataProvider: It is annotation used by TestNG to execute the test method multiple numbers of times based on the data provided by the DataProvider.

@Factory: It is annotation used by the TestNG to execute the test methods present in the same test class using different instances of the respective class.

**What is the return type of @Factory annotation?**

This annotation returns an array of class objects (Object [ ]).

**What are the attributes supported by @Test annotation in TestNG?**

* @Test annotation supports lots of attributes that we can use with this annotation.
* Some of the important attributes supported by Test annotation are alwaysRun, dataProvider, dataProviderClass, dependsOnGroups, dependsOnMethods, enabled, priority, timeOut, etc.

**What is verbose in TestNG?**

verbose is an attribute in TestNG which is mostly used when reporting a bug or when trying to diagnose the execution of test run.

**Jenkins file**

pipeline

{

agent any

stages

{

stage('Build')

{

steps

{

echo "Build is Started"

bat "mvn clean package -PRegression -DskipTests=true"

echo "Build is Successful"

}

}

stage('SonarQube Analysis')

{

steps

{

echo "SonarQube Test is Started"

bat 'mvn sonar:sonar -Dsonar.projectName=MavenHybridFramework -Dsonar.host.url=http://localhost:9000 -Dlicense.skip=true'

echo "SonarQube Test is Successful"

}

}

stage('Smoke TestSuite')

{

parallel

{

stage('Chrome')

{

steps

{

echo "Smoke Test Execution is Started in Chrome"

bat "mvn test -PSmoke -DBrowser=Chrome"

echo "Smoke Test Execution is Successful in Chrome"

}

}

stage('Firefox')

{

steps

{

echo "Smoke Test Execution is Started in Firefox"

bat "mvn test -PSmoke -DskipTests=true"

echo "Smoke Test Execution is Successful in Firefox"

}

}

}

}

stage('Regression TestSuite')

{

parallel

{

stage('Chrome')

{

steps

{

echo "Regression Test Execution is Started in Chrome"

bat "mvn test -PRegression -DBrowser=Chrome"

echo "Regression Tests is Successful in Chrome"

}

}

stage('Firefox')

{

steps

{

echo "Regression Test Execution is Started in Firefox"

bat "mvn test -PRegression -DskipTests=true"

echo "Regression Test Execution is Successful in Firefox"

}

}

}

}

stage('Publish Reports')

{

parallel

{

stage('Extent Report')

{

steps

{

publishHTML([

allowMissing: false,

alwaysLinkToLastBuild: true,

keepAll: false,

reportDir: 'D:\\Automation\_Workspace\\MavenHybridFramework\\CRMExtentResults\\',

reportFiles: 'CRMExtentReport\*.html',

reportName: 'Extent HTML Report',

reportTitles: ''])

}

}

stage('Allure Report')

{

steps

{

echo "Allure Report is yet to be implemented"

}

}

}

}

stage('Notifications')

{

parallel

{

stage('Slack')

{

steps

{

slackSend channel: 'test-automation',

color: 'good',

message: "\*${currentBuild.currentResult}:\* Job Name: ${env.JOB\_NAME} || Build Number: ${env.BUILD\_NUMBER}\n More information at: ${env.BUILD\_URL}"

}

}

stage('Gmail')

{

steps

{

emailext body: "\*${currentBuild.currentResult}:\* Job Name: ${env.JOB\_NAME} || Build Number: ${env.BUILD\_NUMBER}\n More information at: ${env.BUILD\_URL}",

subject: 'Test Automation Pipeline Build Status',

to: 'Pavankrishnan1993@gmail.com'

}

}

}

}

}

post

{

failure

{

echo 'This Job is Failed - Notifications have been sent to Slack and Gmail..!!'

slackSend channel: 'test-automation',

color: 'good',

message: "\*${currentBuild.currentResult}:\* Job Name: ${env.JOB\_NAME} || Build Number: ${env.BUILD\_NUMBER}\n More information at: ${env.BUILD\_URL}"

emailext body: "\*${currentBuild.currentResult}:\* Job Name: ${env.JOB\_NAME} || Build Number: ${env.BUILD\_NUMBER}\n More information at: ${env.BUILD\_URL}",

subject: 'Test Automation Pipeline Build Status',

to: 'Pavankrishnan1993@gmail.com'

}

unstable

{

echo 'This Job is Unstable - Notifications have been sent to Slack and Gmail..!!'

slackSend channel: 'test-automation',

color: 'good',

message: "\*${currentBuild.currentResult}:\* Job Name: ${env.JOB\_NAME} || Build Number: ${env.BUILD\_NUMBER}\n More information at: ${env.BUILD\_URL}"

emailext body: "\*${currentBuild.currentResult}:\* Job Name: ${env.JOB\_NAME} || Build Number: ${env.BUILD\_NUMBER}\n More information at: ${env.BUILD\_URL}",

subject: 'Test Automation Pipeline Build Status',

to: 'Pavankrishnan1993@gmail.com'

}

}

}

### Page Object Model and Page Factory

* Page Object Model (POM) is a design pattern in Selenium that creates an object repository for storing all web elements.
* It is useful in reducing code duplication and improves test case maintenance.
* In Page Object Model, consider each web page of an application as a class file. Each class file will contain only corresponding web page elements.
* Using these elements, testers can perform operations on the website under test.

**Advantages of POM**

1. **Object Repository**: You can create an Object Repository of the fields segmented page-wise. Each page will be defined as a java class. All the fields in the page will be defined in an interface as members. The class will then implement the interface.
2. **Functional Encapsulation**: All possible functionality or operations that can be performed on a page can be defined and contained within the same class created for each page. This allows for clear definition and scope of each page's functionality.
3. **Low maintenance**: Any UI changes can swiftly be implemented into the interface as well as class.
4. **Programmer Friendly**: Robust and more readable. The Object-oriented approach makes the framework programmer friendly.
5. **Low Redundancy**: Helps reduce duplication of code. If the architecture is correctly and sufficiently defined, the POM gets more done in less code.
6. **Efficient & Scalable**: Faster than other keyword-driven/data-driven approaches where Excel sheets are to be read/written.

**What is Page Factory in Selenium?**

* Page Factory is a class provided by Selenium WebDriver to support Page Object Design patterns.
* In Page Factory class all the elements will be initialized at the time of page load.
* initElements() – It is a static method in Page Factory class. Using this method, all the page objects of Page Object class are initialized. It does not mean webdriver locates all web elements of page and store it. We use @FindBy annotation.
* Lazy initialization - AjaxElementLocatorFactory is a lazy load concept in Page Factory. This is used to identify web elements only when they are used in any operation or activity. The timeout of a web element can be assigned to the object class with the help of the AjaxElementLocatorFactory.

@FindBy: Annotation used in Page Factory to locate & declare web elements using different locators.

Below is an example of declaring an element using @FindBy

@FindBy(id="elementId") WebElement element;

Similarly, one can use @FindBy with different location strategies ClassName, Css,Name,Xpath,

TagName,LinkText,PartialLinkText to find the web elements and perform actions on them.

|  |  |
| --- | --- |
| Page Object Model | Page Factory |
| Finding web elements using By | Finding web elements using @FindBy |
| POM does not provide lazy initialization | Page Factory does provide lazy initialization |
| Page Object Model is a design pattern | PageFactory is a class which provides implementation of Page Object Model design pattern |
| In POM, one needs to initialize every page object individually | In PageFactory, all page objects are initialized by using the initElements() method |

<https://github.com/PavanReddy77/MavenHybridFramework>

<https://github.com/ivodimitrov/selenium-webdriver-java-framework>

<https://github.com/naveenanimation20/PageObjectModel>

<https://github.com/hverma22/MyProject>

<https://github.com/nadvolod/sc-20>

<https://github.com/GopinathJayakumar>

## How to Explain Automation Framework during Interview

<https://youtu.be/KLdQ493xWMk>

Creating a Production Level Automation Framework in Under 60 Minutes

<https://youtu.be/mvmduwi2WWA>

Framework Interview Questions

<https://www.facebook.com/groups/570227323001088/permalink/3604517486238708/?sfnsn=wiwspmo&extid=0lLhzwUZ9JtNV76k>

<https://www.lambdatest.com/blog/selenium-best-practices-for-web-testing/>

<http://www.softwaretestingportal.com/>

<https://www.automationtestinginsider.com/>

**Verify Image Presence in Web Page using Selenium WebDriver**

@Test

public void CheckImage() throws Exception {

driver.get(baseUrl);

WebElement ImageFile = driver.findElement(By.xpath("//img[contains(@id,'Test Image')]"));

Boolean ImagePresent = (Boolean) ((JavascriptExecutor)driver).executeScript("return

arguments[0].complete && typeof arguments[0].naturalWidth != \"undefined\" && arguments[0].naturalWidth > 0", ImageFile);

if (!ImagePresent)

{

System.out.println("Image not displayed.");

}

else

{

System.out.println("Image displayed.");

}

}

**Find Broken / Invalid Images on a Page**

* There are cases where we have seen image loading is failed due to many reasons, the most see is "Image not loading - Failed to load the given URL” because of image file is not located in the same location as that is specified or may be image file is corrupted.
* And it will be very difficult to identify invalid images when there are many in the applications.
* To achieve this, we can use HTTPClient library to check status codes of the images on a page. If they don't load correctly, then it will be registered with likely a 404 but not a 200 status code.
* We can easily say whether the link is broken or not with status codes.

import java.util.List;

import org.apache.http.HttpResponse;

import org.apache.http.client.HttpClient;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.impl.client.HttpClientBuilder;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class FindBrokenImages {

private WebDriver driver;

private int invalidImageCount;

@BeforeClass

public void setUp() {

driver = new FirefoxDriver();

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

}

@Test

public void validateInvalidImages() {

try {

invalidImageCount = 0;

List<WebElement> imagesList = driver.findElements(By.tagName("img"));

System.out.println("Total no. of images are " + imagesList.size());

for (WebElement imgElement : imagesList) {

if (imgElement != null) {

verifyimageActive(imgElement);

}

}

SOP("Total no. of invalid images are " + invalidImageCount);

} catch (Exception e) {

e.printStackTrace();

System.out.println(e.getMessage());

}

}

@AfterClass

public void tearDown() {

if (driver != null)

driver.quit();

}

public void verifyimageActive(WebElement imgElement) {

try {

HttpClient client = HttpClientBuilder.create().build();

HttpGet request = new HttpGet(imgElement.getAttribute("src"));

HttpResponse response = client.execute(request);

// verifying response code he HttpStatus should be 200 if not,

// increment as invalid images count

if (response.getStatusLine().getStatusCode() != 200)

invalidImageCount++;

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Difference between get() and navigate() methods in selenium**

|  |  |
| --- | --- |
| get() | navigate() |
| Web Driver will wait until the page has fully loaded (that is, the onLoad event has fired) before returning control to your test or script. It’s worth noting that if your page uses a lot of AJAX on load then Web Driver may not know when it has completely loaded. If you need to ensure such pages are fully loaded then you can use waits. | This method navigates to an URL and It will not wait till the whole page gets loaded. |
| It does not maintain the browser history and cookies so can’t use back or forward. | It maintains browser history or cookies to navigate back or forward. |

**Why do we need Abstract and Interfaces in Selenium**

* Selenium WebDriver is an interface that defines a set of methods.
* However, implementation is provided by the browser specific classes.
* Some of the implementation classes are AndroidDriver, ChromeDriver, FirefoxDriver, InternetExplorerDriver, IPhoneDriver, SafariDriver etc.

The WebDriver main functionality is to control the browser. It even helps us to select the HTML page elements and perform operations on them such as click, filling a form fields etc.

**Selenium WebDriver Methods**

SearchContext is the topmost interface in Selenium API which has two methods – findElement() and findElements().

Selenium WebDriver interface has many abstract methods like get(String url), quit(), close(), getWindowHandle(), getWindowHandles(), getTitle() etc.

WebDriver has nested interfaces like Window, Navigation, Timeouts etc. These nested interfaces are used to perform operations like back(), forward(), refresh(), timeOut(), maximize(), minimize().

**List of classes implementing WebDriver**

The major implementation classes of WebDriver interface are ChromeDriver, EdgeDriver, FirefoxDriver, InternetExplorerDriver etc. Each driver class corresponds to a browser. We simply create the object of the driver classes and work with them.

**List of Commands on WebElement** - findElement(), sendKeys(), clear(), click()

**Selenium Web Driver Hierarchy and Architecture - Brief Overview**

**JSON Wire Protocol**

* Selenium Web Driver uses JSON Wire Protocol to communicate with browser in request/response pairs of "commands" and "responses".
* This wire protocol defines a RESTful web service using JSON over HTTP.
* Your client i.e. your automated scripts send commands to your server i.e. browser and receives Reponses accordingly.
* The main Benefits of Selenium Web Driver using JSON Wire Protocol is that you can write your code in any programming language and your commands with HTTP verbs go to the browser and receives response in Json.
* Other benefits include running your automated scripts against any kind of browser with the same implementation, running your scripts on cloud service providers like SauceLabs, BrowserStack, Perfecto cloud etc.
* For Example; getTitle(), getText(), getPageSource() methods are GET requests, click(), findElement() methods are POST requests.
* The tables describe some common status codes returned by server i.e. browser to the user:

|  |  |
| --- | --- |
| **Status Code** | **Status Description** |
| 0 | Success |
| 7 | NoSuchElement |
| 8 | NoSuchFrame |
| 10 | StaleElementReference |
| 11 | ElementNotVisible |
| 15 | ElementIsNotSelectable |
| 23 | NoSuchWindow |
| 32 | InvalidSelector |

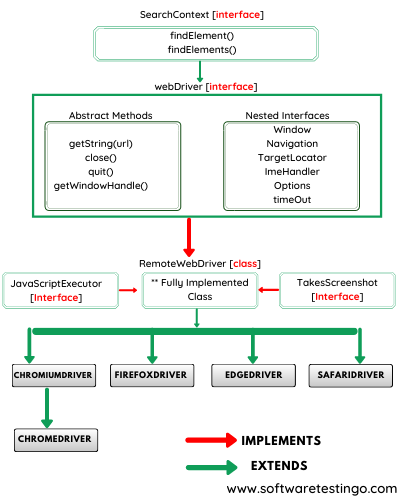
* To give you a brief idea of JSON request and response body, I will give you one example of GET requests of getTitle() method and its response body.
* Your request to fetch title of the current window is sent to this endpoint /session/:sessionId/title where ‘sessionId’ is url parameter and value of your browser session.
* It ‘s request sends desired capabilities in the JSON body so request JSON looks like this for Chrome browser:

{"desiredCapabilities" : {"browserName" : "chrome"}}

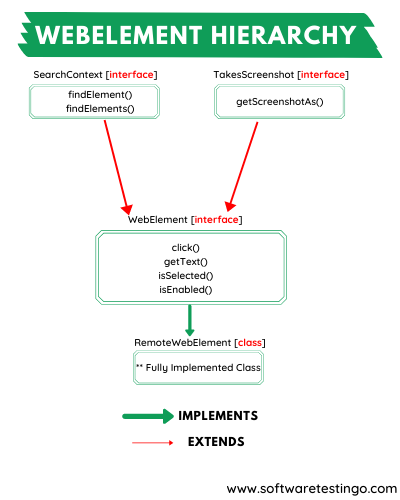
* It returns a status code in response as well as value of the title assuming it’s success so it’s response Json looks like this

{"status" : 0, "value" : "xyz"}

**Selenium Web driver Interface Hierarchy**



**Web Element Hierarchy**



**Difference between Web Services and API**

|  |  |
| --- | --- |
| Web Services | API |
| A Web service is a way for two machines to communicate with each other over a network. | API is a set of definitions and protocols that allow one application to communicate with another application without any user involvement. |
| Web service is used for REST, SOAP and XML-RPC for communication. | API is used for any style of communication. |
| Web service supports only HTTP protocol. | API supports HTTP/HTTPS protocol: URL Request/Response Headers, etc. |
| Web service supports XML. | API supports XML,JSON ,Text formats. |
| All Web services are APIs. | But all APIs are not web services. |

**1. Log4j**

* Add/Create Log4j.xml in project directory
* Add/Create Log Class in utility Package
* Configure @BeforeSuite at BaseClass to configure log4j.xml
* DOMConfigurator.configure("log4j.xml");
* Need to just Call in methods in testCase from Log class

**2. DataDriven Testing and DataProvider**

* Add/Create ExcelLibrary in utility package.
* Create a Folder and add TestData.xlxs in that.
* Create a package for DataProvider and add DataProvider class there
* and create the object of ExcelLibrary Class
* Add the DataProvider methods
* Call the DataProvider methods from testcases

**3. Extent Report**

* Add/Create extent-config.xml file for Extent Report Configuration
* Add/Create ExtentManager Class in utility Package-- to create the object
* of ExtentHtmlReporter and load extent-config.xml
* Create a folder ro Save Extent Report under test-output
* Configure ExtentManager.setExtent() in @BeforeSuite method in BaseClass
* Configure ExtentManager.endReport() in @AfterSuite method in BaseClass
* Add/Create screenShot method in Action/BaseClass

**To attach the screenshot in extent report**

* Add/Create a Listener Class -- ListenerClass
* To call the listener Add the below listener (inside suite tag)
* setting in testng.xml

<listeners>

<listener class-name="com.Project.util.ListenerClass"></listener>

</listeners>

**Building a Selenium Framework from A to Z**

<https://www.logigear.com/blog/test-automation/building-a-selenium-framework-from-a-to-z/>

**How to build a maintainable Selenium framework?**

1. Choose a programming language
2. Choose a unit test framework
3. Design the framework architecture
4. Build the SeleniumCore component
5. Build the SeleniumTest component
6. Choose a reporting mechanism
7. Decide how to implement CI/CD
8. Integrate your framework with other tools

<https://demo.nopcommerce.com>

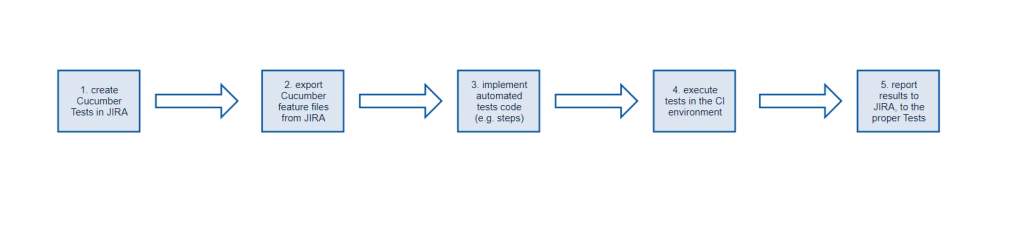
<https://admin-demo.nopcommerce.com>

<https://total-qa.com/selenium-webdriver/demo-sites/>

**Overview**

1. The BA (Business Analyst) will give the requirement to the test manager.
2. The test manager will create new cucumber tests then create test execution on Jira.
3. Testers will download cucumber files to the automation project then they will develop tests and commit the code to source control like Github or SVN,…
4. After that, they will create and configure jobs for these tests on Jenkins to execute tests.
5. According to our model, after execution, the report will be imported back to Xray (Jira) – a test management tool.
6. All tests are running in headless browsers (PhantomJS (old), Chrome or Firefox headless mode,…).
7. In addition, the test not only runs on localhost it also runs a cloud node (such as Saucelabs).

In this workflow, Tests are created and managed in Jira, thus Jira will be the master for the Cucumber scenarios.



1. Specify Cucumber tests using natural language, in Jira.
2. Export Cucumber features from Jira to the CI environment, using the [Xray REST API](https://confluence.xpand-addons.com/display/XRAY/Import+Execution+Results+-+REST).
3. Implement tests in code and commit them to the source code versioning system.
4. Execute tests in the CI environment.
5. Report results to Xray, using the [Xray REST API](https://confluence.xpand-addons.com/display/XRAY/Import+Execution+Results+-+REST).

# [Integrating Selenium Tests with Jira Step-by-Step Guide](https://codelabslk.com/integrating-selenium-tests-with-jira/)

<https://codelabslk.com/integrating-selenium-tests-with-jira/>

The complete source code for this tutorial can be found at

<https://github.com/codelabslk/selenium-jira-integration-example>

**Database Testing using Selenium**

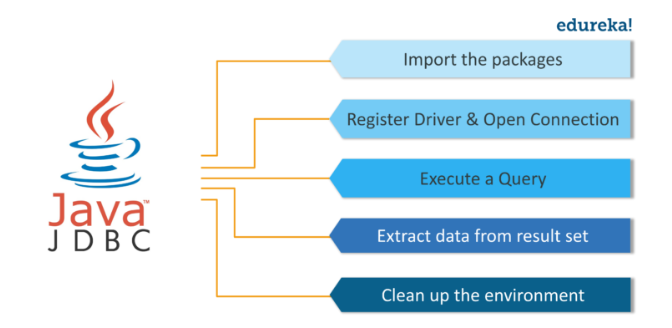
Common JDBC Components

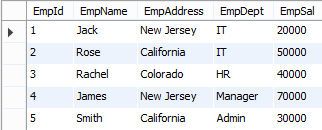
The JDBC API provides the following interfaces and classes −

DriverManager: It is used to manage a list of database drivers. This driver recognizes a certain subprotocol under JDBC in order to establish a database Connection.

Driver: It is an interface that handles the communications with the database server.

Connection: It is an interface that consists of all the methods required to connect to a database. The connection object represents communication context wherein the entire communication with the database is through connection object only.





import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class DatabaseTesingDemo {

static Connection con = null;

private static Statement stmt;

public static String DB\_URL = "jdbc:mysql://localhost:3306/easy";

public static String DB\_USER = "root";

public static String DB\_PASSWORD = "root";

@BeforeClass

public void setUp() throws Exception {

try{

Class.forName("com.mysql.jdbc.Driver");

Connection connect = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);

if (connection != null) {

System.out.println("Connected to the Database...");

}

} catch (SQLException ex) {

ex.printStackTrace();

}

catch (ClassNotFoundException ex) {

ex.printStackTrace();

}

}

@Test

public void testEmpFromDB() {

try {

String query = "select \* from employee";

stmt = con.createStatement();

ResultSet rs = stmt.executeQuery(query);

while (rs.next())

{

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

String EmpName = rs.getStrig("EmpName");

String EmpAddress=rs.getString(3);

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

Double EmpSal= rs.getDouble(5);

}

}

catch(Exception e) {

e.printStackTrace();

}

}

@AfterTest

public void tearDown() throws Exception {

if (con != null) {

try {

System.out.println("Closing Database Connection...");

connection.close();

} catch (SQLException ex) {

ex.printStackTrace();

}

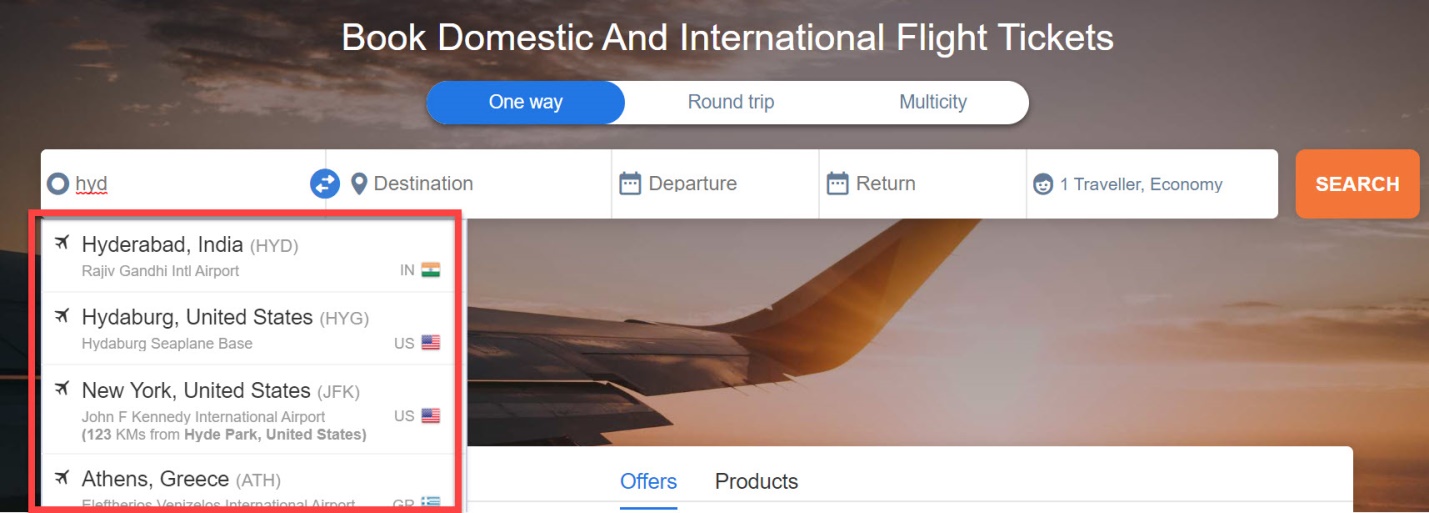
}

}

}

**How to select an option from auto suggest dropdown in selenium.**

**Example of auto suggest dropdown:**



**import static org.testng.Assert.fail;**

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

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

**import org.openqa.selenium.JavascriptExecutor;**

**import org.openqa.selenium.Keys;**

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

**import org.openqa.selenium.chrome.ChromeDriver;**

**public class AutoSuggestionDropdown {**

**static WebDriver driver = null;**

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

**AutoSuggestionDropdown auto = new AutoSuggestionDropdown();**

**auto.suggestions();**

**auto.closeBrowser();**

**}**

**public void suggestions() throws Exception{**

**System.setProperty(“webdriver.chrome.driver”, “C:\\chromedriver\_win32\\chromedriver.exe”);**

**driver=new ChromeDriver();**

**driver.get(“https://global.shopbakerhughes.com/”);**

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

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

**Thread.sleep(2000);**

**driver.findElement(By.xpath(“//input[@id=’search’]”)).sendKeys(“drill”);**

**driver.findElement(By.xpath(“//input[@id=’search’]”)).sendKeys(Keys.DOWN);**

**// It doesn’t print anything, because text in the search bar is invisible until we select it.**

**String text = driver.findElement(By.xpath(“//input[@id=’search’]”)).getText();**

**System.out.println(“Text is “+text);**

**Thread.sleep(2000);**

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

**String str = “return document.getElementById(\”search\”).value;”;**

**String item = (String) js.executeScript(str);**

**System.out.println(item);**

**int i=0;**

**while(!item.contains(“drilling fluid”)){**

**i++;**

**driver.findElement(By.xpath(“//input[@id=’search’]”)).sendKeys(Keys.DOWN);**

**Thread.sleep(1000);**

**item = (String) js.executeScript(str);**

**System.out.println(item);**

**if(i>15){**

**break;**

**}**

**}**

**if(i>15){**

**System.out.println(“Item not found.”);**

**fail(“Desired item not found.”);**

**}else{**

**System.out.println(“Desire item found.”);**

**}**

**driver.findElement(By.xpath(“//input[@id=’search’]”)).sendKeys(Keys.TAB);**

**driver.findElement(By.xpath(“//button[@title=’Search’]”)).click();**

**}**

**public void closeBrowser(){**

**driver.close();**

**}**

**}**

**HANDLE BOOTSTRAP DROPDOWN (WITHOUT SELECT CLASS) IN SELENIUM**

**import java.util.List;**

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

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

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

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

**import org.openqa.selenium.chrome.ChromeDriver;**

**import org.testng.annotations.Test;**

**public class BootStrapDropDown\_Reusable**

**{**

**@Test**

**public void BootStrapDropDown\_Reusables()**

**{**

**WebDriver driver;**

**System.setProperty("webdriver.chrome.driver", "C:\\Vision\\chromedriver.exe"); driver = new ChromeDriver();**

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

**driver.get("https://www.jquery-az.com/boots/demo.php?ex=63.0\_2");**

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

**// Clicking on Bootstrap Dropdown**

**driver.findElement(By.xpath("//button[contains(@class,'multiselect')]")).click();**

**// Get the all WebElements inside the dropdown in List**

**List<WebElement> dropdown\_list = driver.findElements(By.xpath("//ul[contains(@class,'multiselect-container dropdown-**

**menu')]//li//a//label"));**

**System.out.println("The Options in the Dropdown are: " + dropdown\_list.size());**

**// Condition to get the WebElement for list and Click over "Angular" option**

**for(int i=0; i<dropdown\_list.size(); i++)**

**{**

**System.out.println(dropdown\_list.get(i).getText());**

**// Checking the condition whether option in text "Angular" is coming**

**if(dropdown\_list.get(i).getText().contains("Angular"))**

**{**

**// Clicking if text "Angular" is there**

**dropdown\_list.get(i).click();**

**break;**

**}**

**}**

**}**

**}**

**How to get to know whether drop down list values are in sorted order or not - Selenium WebDriver**

**import static org.junit.Assert.\*;**

**import java.util.ArrayList;**

**import java.util.List;**

**import org.junit.Test;**

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

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

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

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

**public class DropDownListSortedOrNot {**

**@Test**

**public void dropDownListSortedOrNot() throws InterruptedException {**

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

**driver.get("http://www.ebay.in/");**

**Thread.sleep(3000);**

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

**element.click();**

**List<WebElement> dropDownvalues = element.findElements(By.tagName("option"));**

**ArrayList<String> listValues = new ArrayList<String>();**

**for(WebElement value : dropDownvalues) {**

**System.out.println("values are"+ value.getText());**

**listValues.add(value.getText());**

**}**

**boolean sortedOrNot = sortedOrNot(listValues);**

**assertEquals(true, sortedOrNot);**

**driver.close();**

**}**

**public boolean sortedOrNot(ArrayList<String> dropDownValues) {**

**System.out.println("number of values "+ dropDownValues.size());**

**for(int i=0; i<dropDownValues.size();i++) {**

**int temp = dropDownValues.get(i).compareTo(dropDownValues.get(i+1));**

**if(temp>1) {**

**System.out.println("i value"+i);**

**return false;**

**}**

**}**

**return true;**

**}**

**}**

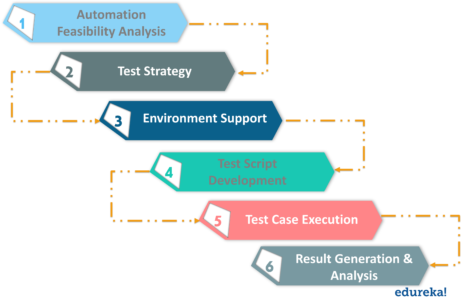
**Example Test Script**

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Operation** | **Object** | **Action** |
| **1** | **Call Action** | **<Name of the excel file that contains test script>** |  |
| **2** | **Perform** | **Link;Manage Projects** | **click** |
| **3** | **Wait** | **3** |  |
| **4** | **Perform** | **ButtonCreate;New Project** | **click** |
| **5** | **Perform** | **TextBox;Project Name** | **Set:Selenium** |

**Telling Selenium where everything is**

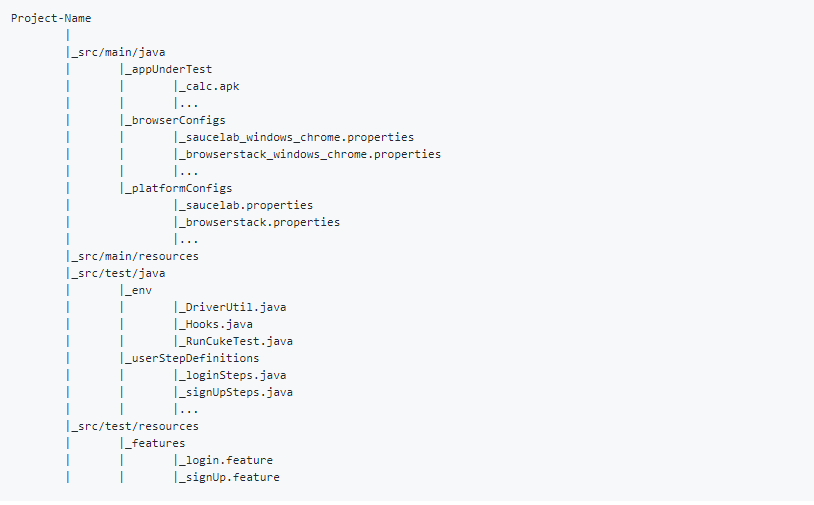
|  |  |
| --- | --- |
| **File/Folder Name** | **Location** |
| **Test Script** | **<Path of Test Script>** |
| **Object Repository** | **<Path of OR>** |
| **Environment File** | **<Path of Environment File>** |
| **Summary Report** | **<Path of Summary Report>** |
| **Screenshot** | **<Path of Screenshot>** |
| **Detailed Report** | **<Path of detailed report>** |

**Test Automation Life Cycle**

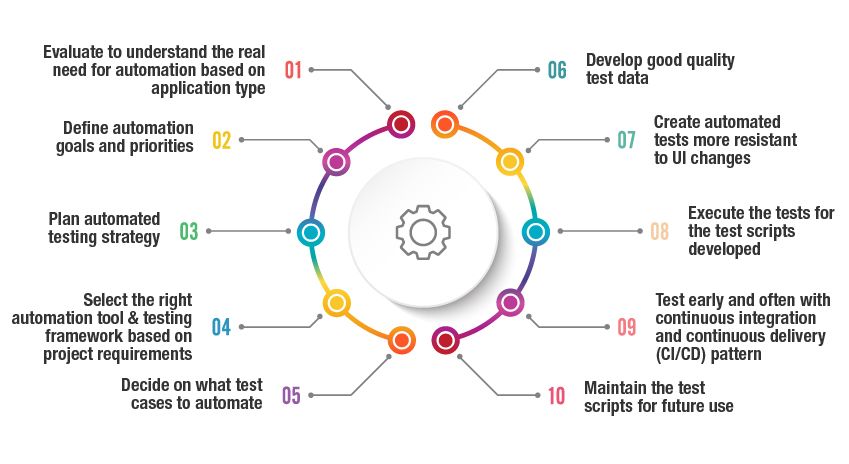


* **Automation Feasibility Analysis:** In this phase, you check the feasibility of automation. This includes short listing the test cases for automation and selecting the right test tool.
* **Test Strategy:**In test strategy, you select the test automation framework. You create a test plan and test automation suite in your test management tool.
* **Environment Set up:**In this phase you set up the testing environment and acquire the required hardware and software to execute the automated test cases.
* **Test Script Development:** In this phase, you start creating the automation test scripts. Make sure that your scripts are reusable, well-structured and well documented.
* **Test Case Execution:**In this phase, you execute your test scripts.
* **Test Result Generation and Analysis:**This is the last phase of test automation life cycle. In this phase, you analyse the output of test cases and share the reports with stakeholders.

**Framework Architecture**



**What are the steps for an effective Test Automation Approach?**



**How to explain Test Automation Framework to the interviewer?**

A framework is defined as a set of rules or best practices that can be followed in a systematic way that ensures to deliver the desired results.

We need to specify in and out of our Test Automation Framework such as programming language used, Type of framework used, Test Base Class (Initializing WebDriver, Implicit Waits), How we separate Element locators and tests (Page Objects, Page Factory), Utility functions file, Property files, TestNG annotations, How we parameterize tests using Excel files, How we capture error screenshots, Generating reports(Extent Reports), Emailing reports, Version Control System used and Continues Integration Tool used.

**Type of Framework**: In our project, we are using Data-driven Framework by using Page Object Model design pattern with Page Factory.

**POM**: As per the Page Object Model, we have maintained a class for every web page. Each web page has a separate class and that class holds the functionality and members of that web page. Separate classes for every individual test.

**Packages**: We have separate packages for Pages and Tests. All the web page related classes come under Pages package and all the tests related classes come under Tests package.

Above screenshot illustrates standardized maven project. As per the above maven project, all the tests are kept in the ‘src/test/java‘ and remaining files (such as config.properties, element locators (POM classes), utility files, test data, etc.,) kept under ‘src/main/java‘.

**Test Base Class**: Test Base class (TestBase.java) deals with all the common functions used by all the pages. This class is responsible for loading the configurations from properties files, Initializing the WebDriver, Implicit Waits, Extent Reports and also to create the object of FileInputStream which is responsible for pointing towards the file from which the data should be read.

**Utility Classes** (AKA Functions Class): Utility class (TestUtil.java) stores and handles the functions (The code which is repetitive in nature such as waits, actions, capturing screenshots, accessing excels, sending email etc.,) which can be commonly used across the entire framework. The reason behind creating utility class is to achieve reusability. This class extends the TestBase class to inherit the properties of TestBase in TestUtil.

**Properties file**: This file (config.properties) stores the information that remains static throughout the framework such as browser specific information, application URL, screenshots path etc.

All the details which change as per the environment and authorization such as URL, Login Credentials are kept in the config.properties file. Keeping these details in a separate file makes easy to maintain.

**Screenshots**: Screenshots will be captured and stored in a separate folder and also the screenshots of a failed test cases will be added in the extent reports.

**Test Data**: All the historical test data will be kept in excel sheet (controller.xlsx). By using this file, we pass test data and handle data driven testing. We use Apache POI to handle excel sheets.

**TestNG**: Using TestNG for Assertions, Grouping and Parallel execution.

**Maven**: Using Maven for build, execution and dependency purpose. Integrating the TestNG dependency in POM.xml file and running this POM.xml file using Jenkins.

**Version Control Tool**: We use Git as a repository to store our test scripts.

**Jenkins**: By using Jenkins CI (Continuous Integration) Tool, we execute test cases on daily basis and also for nightly execution based on the schedule. Test Result will be sent to the peers using Jenkins.

**Extent Reports**: For the reporting purpose, we are using Extent Reports. It generates beautiful HTML reports. We use the extent reports for maintaining logs and also to include the screenshots of failed test cases in the Extent Report.

**How to crack Selenium WebDriver Interview without Experience**

<https://www.youtube.com/watch?v=mS_U0ibgQ7s>

<https://github.com/letskodeit/selenium-webdriver-interviews>

<http://www.pavantestingtools.com/p/selenium.html>

**Good Test Automation Framework Checklist**

1. Reusable methods or page classes – Create reusable methods wherever you find repeatable code. Don’t duplicate the same thing in multiple tests.
2. Data driven – Test data like URLs / User Names and Passwords should be maintained in properties file or Excel files. Don’t hard code everywhere.
3. Explicit waits – Thread sleep delays everywhere in test scenarios. Also reduce the performance. So try to use explicit waits.
4. Variables names should be meaningful.
5. Try to use public API’s instead of creating more utility files from scratch.
6. Reporting – Don’t print results using System.out.println. Always use Reporting mechanisms.
7. Headless test execution support when there is a necessity.
8. Don’t hard code absolute paths given to files used in the framework; instead use the files into a folder relative to the framework.
9. Data should read from test scenarios but not in page classes.
10. Try to reduce unnecessary program loops in the code.
11. Test framework should organize into well-defined packages

Pages – Where page classes reside

Test – Where test reside

Utility – Where utility classes resides. Such as reporting and file reading classes

1. Documentation on deploying the test framework.
2. Logging facility for frameworks, when something goes wrong.
3. Base driver support to run in multiple browsers.
4. Good naming conventions for page class and test class naming.
5. Tests should be independent when executing.
6. Detailed reports on test executions and failures.
7. Use design patterns and principals.
8. Use BDD – But this is not mandatory always.
9. Screen shots on failures - Helps failure investigation easy.
10. Use dependency management like Maven for Java, Nuget for .net, PIP for Python

**Verifying whether an element present or visible in selenium Webdriver**

**To check Element Present**

**if(driver.findElement(By.xpath("value"))!= null) {**

**System.out.println("Element is Present");**

**}else{**

**System.out.println("Element is Absent");**

**}**

**To check Visible**

**if( driver.findElement(By.cssSelector("a > font")).isDisplayed()) {**

**System.out.println("Element is Visible");**

**}else{**

**System.out.println("Element is InVisible");**

**}**

**To check Enable**

**if( driver.findElement(By.cssSelector("a > font")).isEnabled()) {**

**System.out.println("Element is Enable");**

**}else{**

**System.out.println("Element is Disabled");**

**}**

**To check text present**

**if(driver.getPageSource().contains("Text to check")) {**

**System.out.println("Text is present");**

**}else{**

**System.out.println("Text is absent");**

**}**

**Handling hidden elements in Selenium**

In case of ID exist for the element, we can use .getElementById javascript method to perform the click operation

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

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

In case there is no id or name assigned to the webelement, still we can interact as given below. Identify the element by any of the standard webdrive By class.

WebDriver element = webdriver.findElement(By.xpath(“<locator>”));

Now the argument[0] will always represent the webelement holding by “element” object in the above statement.

JavascriptExecutor Executor = (JavascriptExecutor)webdriver);

Executor.executeScript(“arguments[0].click();”, element);

We can even perform the action on the specified elements by making it to visible during run time of the automation script. Generally in web application, elements visibility is controlled majorly using the CSS (which are attached to webpages) attributes like style=”display: none;”

We can change the style attribute or even add new attributes which can cause a temporary change in the application look and feel which helps to execute selenium automation scripts uninterruptedly.

Below, we have changed the style property as style=”visibility : visible; which has made the drop down or combo box as visible for selenium WebDriver interactions. we can use the javascript to change the attribute values dynamically.

Executor.executeScript(“arguments[0].setAttribute(‘style’, ‘visibility: visible;’);”, element);

**Example to handle hidden element**

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class Herokuapp {

public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver","C:\\Program Files (x86)\\Hexaware\\STM Client\\Drivers\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.manage().deleteAllCookies();

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

driver.get("http://the-internet.herokuapp.com/");

driver.findElement(By.linkText("Add/Remove Elements")).click();

//driver.findElement(By.xpath("//button[text()='Add Element']")).click();

List<WebElement> deleteButtons = driver.findElements(By.xpath("//div[@class='example']/child::div[1]/button"));

System.out.println(deleteButtons.size());

if(deleteButtons.size()!=0) {

System.out.println("The delete buttons are avalible in the DOM");

for(int i =0;i<deleteButtons.size();i++) {

System.out.println(i + "delete button is getting clicked");

deleteButtons.get(i).click();

}

} else {

}

System.out.println("The delete buttons in the web page are disabled");

}

}

<https://automation-seleniumtutorial.blogspot.com/2019/01/how-to-handle-hidden-elements-in-selenium-webdriver.html>

**Explain how you will login into any site if it is showing any authentication popup for username and password?**

Since there will be popup for logging in, we need to use the explicit command and verify if the alert is actually present. Only if the alert is present, we need to pass the username and password credentials.

The sample code:

WebDriver wait = new WebDriverWait(driver, 10);

Alert alert = wait.until(ExpectedConditions.alertIsPResent());

alert.authenticateUsing(new UserAndPassword(\*\*username\*\*, \*\*password\*\*));

**How to handle HTTP Proxy Authentication Popup in Selenium?**

**Syntax :** [**http://username:password@URL**](http://username:password@URL)

driver.get(<https://admin:admin@the-internet-herokuapp.com/basic_auth>)

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

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

for (int i = 0; i < links.size(); i++) {

WebElement element = links.get(i);

String url = element.getAttribute("href");

URL link = new URL(url);

HttpURLConnection httpConn = (HttpURLConnection) link.openConnection();

httpConn.setConnectTimeout(2000);

httpConn.connect();

if (httpConn.getResponseCode() >= 400) {

System.out.println(url + " - " + "is Broken Link");

} else {

System.out.println(url + " - " + "is valid Link");

}

**How to click on an element which is not visible using selenium WebDriver?**

We can use JavascriptExecutor to click.

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

JavascriptExecutor executor = (JavascriptExecutor)driver;

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

**How many test cases you automated per day?**

* It depends on Test case scenario complexity and length.
* I did automate 2-5 test scenarios per day when the complexity is limited.
* Sometimes just 1 or fewer test scenarios in a day when the complexity is high.

**How did you run automation in 2-3 browsers?**

Selenium Grid. Some companies use One VM, one browser at a time. Otherwise we ran automation scripts in IE on Monday, Chrome on Tuesday and FF on Wednesday.

**Which Framework have you used to execute test Cases?**

TestNG

**How long your automation execution takes to complete and how you reduced the time?**

* Sanity Test Suite around 1 hour || Regression Test Suite 2 to 10 hours.
* To reduce the time I used TestNG Parallel feature which were not dependent on each other.

**How do you debug failed test cases?**

We can look at the Logs and Screenshots to debug the failed test cases.

**How do you perform logging and how you take screenshots?**

* Log4j for logging purpose.
* In testing, Implement ITestListener=>onTestFailure() method, swhere we implement our code to take the screenshot.

**What is a hub in Selenium Grid?**

A hub is a server or a central point that controls the test executions on different machines.

**What is a node in Selenium Grid?**

Node is the machine which is attached to the hub. There can be multiple nodes in Selenium Grid.

**What is the super interface of Web Driver?**

SearchContext

**List some scenarios which we cannot automate using Selenium Web Driver?**

* Bitmap comparison Is not possible using Selenium Web Driver
* Automating Captcha is not possible using Selenium Web Driver
* We can not read bar code using Selenium Web Driver
* windows OS based pop ups
* third party calendars/element
* Image
* Word/PDF

**How can you use the Recovery Scenario in Selenium Web Driver?**

By using “Try Catch Block” within Selenium Web Driver Java tests.

try {

driver.get("www.xyz.com");

}catch(Exception e){

System.out.println(e.getMessage());

}

**How to schedule the Test Suite Execution?**

We can schedule the test suite execution using CI tools like Jenkins, Bamboo. Alternatively, we can use windows scheduler to launch the test execution.

**How do you make sure that a failure is not intermittent?**

While executing the automation scripts, test cases may fail for several reasons. To optimize our next runs, we need to re-run only failed test cases. How to Execute failed Test cases? What is the best approach? In TestNg class, we can easily re-run the test cases using two methods as explained below:

Method 1: By using testng-failed.xml file in test-output folder.

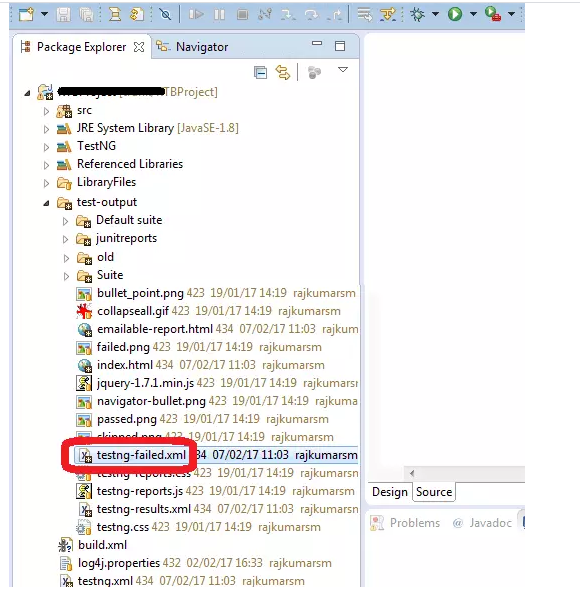
Method 2: By implementing TestNG IRetryAnalyzer.

**Method 1: Execute failed test cases using TestNG in Selenium – By using “testng-failed.xml”**

**Steps To follow**

* Once the execution is done, automatically test-output folder will be created in the project folder.
* In the test-output folder >> testng-failed.xml file will be created.
* Right click on testng-failed.xml>> Run as TestNg. This will now only re-run the fail test case(s).

Disadvantage here is we can’t execute single fail testcase multiple times. For that in second method we can easily retry multiple time fail testcases.



**Method 2: By implementing TestNG IRetryAnalyzer**

There are many reasons for testcases to fail. Some of them could be:

Element not visible,

Stale element,

Time out exception etc.

If testcases have failed for above reasons, then we can easily re-run such testcases in TestNg by implementing the IRetryAnalyzer class. The steps are explained further below.

import org.testng.IRetryAnalyzer;

import org.testng.ITestResult;

public class RetryFailedTestCases implements IRetryAnalyzer {

private int retryCnt = 0;

//You can mention maxRetryCnt (Maximium Retry Count) as per your requirement.

private int maxRetryCnt = 2;

//This method will be called everytime a test fails. It will return TRUE if a test fails and need to be retried, else it returns FALSE

public boolean retry(ITestResult result) {

if (retryCnt < maxRetryCnt) {

System.out.println(“Retrying ” + result.getName() + ” again and the count is ” + (retryCnt+1));

retryCnt++;

return true;

}

return false;

}

}

Create another class RetryListenerClass to implement ‘IAnnonation Transformer’ interface. The implementation of ‘IAnnotationTransformer’ interface may help us set the ‘setRetryAnalyzer’ for ‘ITestAnnotation.

import java.lang.reflect.Constructor;

import java.lang.reflect.Method;

import org.testng.IAnnotationTransformer;

import org.testng.IRetryAnalyzer;

import org.testng.annotations.ITestAnnotation;

public class RetryListenerClass implements IAnnotationTransformer {

@Override

public void transform(ITestAnnotation testannotation, Class testClass, Constructor testConstructor, Method testMethod) {

IRetryAnalyzer retry = testannotation.getRetryAnalyzer();

if (retry == null) {

testannotation.setRetryAnalyzer(RetryFailedTestCases.class);

}

}

}

Create one sample class with two test methods:

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class TestNGExampleTests {

WebDriver driver;

String baseURL = “https://www.gmail.com/”;

@BeforeClass

public void setup() {

System.setProperty(“webdriver.chrome.driver”, “G:\\SampleWorkSpace\\Sample\\Jar\\chromedriver.exe”);

driver = new ChromeDriver();

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

}

@Test(priority=1)

public void verifycurrentUrl() {

driver.navigate().to(baseURL);

String ActualResult=driver.getCurrentUrl();

System.out.println(ActualResult);

String ExceptedResult = “https://accounts.google.com/signin/v2/identifier?service=mail&passive=true&rm=false&continue=https%3A%2F%2Fmail.google.com

%2Fmail%2F&ss=1&scc=1&ltmpl=default&ltmplcache=2&emr=1&osid=1&flowName=GlifWebSignIn&flowEntry=ServiceLogin”;

Assert.assertEquals(ActualResult, ExceptedResult);

}

@Test(priority=2)

public void VerifyCurrentURl() throws InterruptedException

{

driver.navigate().to(baseURL);

Thread.sleep(10000);

driver.findElement(By.linkText(“Sign in”)).click();

Thread.sleep(5000);

WebElement element = driver.findElement(By.xpath(“/html/body/div[1]/div[1]/div[2]/div[2]/div/div/div[2]/div/div[1]/div/form/span/section/div/div/div [3]/button”));

String ActualResult = element.getText();

Assert.assertEquals(ActualResult, “Forgot email?”);

}

}

In the above class, Test 2 method is getting failed and it will retry 2 times in Retry class. We need to add below code in testng.xml file.

<listeners>

<listener class-name=”com.RetryListener”/>

</listeners>

Sample testng.xml file:

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

<suite name=”Sample Test Suite” verbose=”1″ >

<listeners>

<listener class-name=”com.RetryListenerClass”/>

</listeners>

<test name=”VerifycurrentUrl()” >

<classes>

<class name=”com.TestNGExampleTests” />

</classes>

</test> <!– Test –>

<test name=”VerifyText()” >

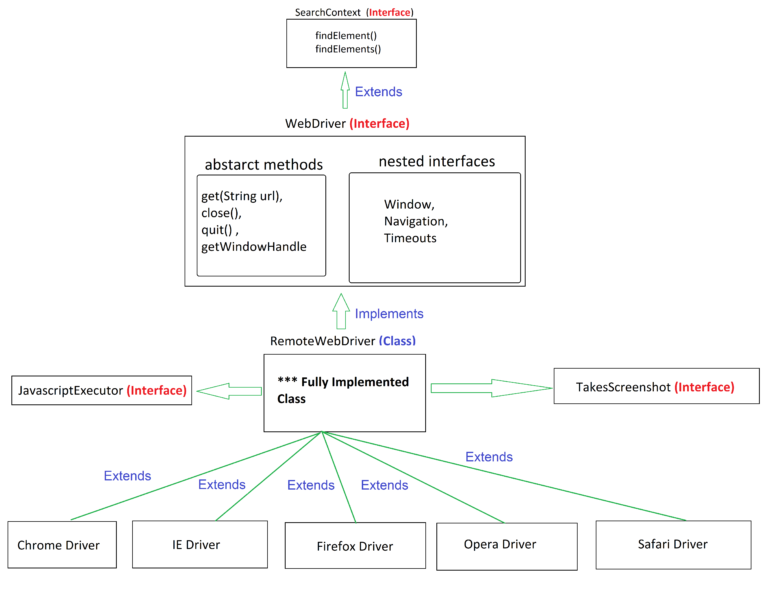
<classes>

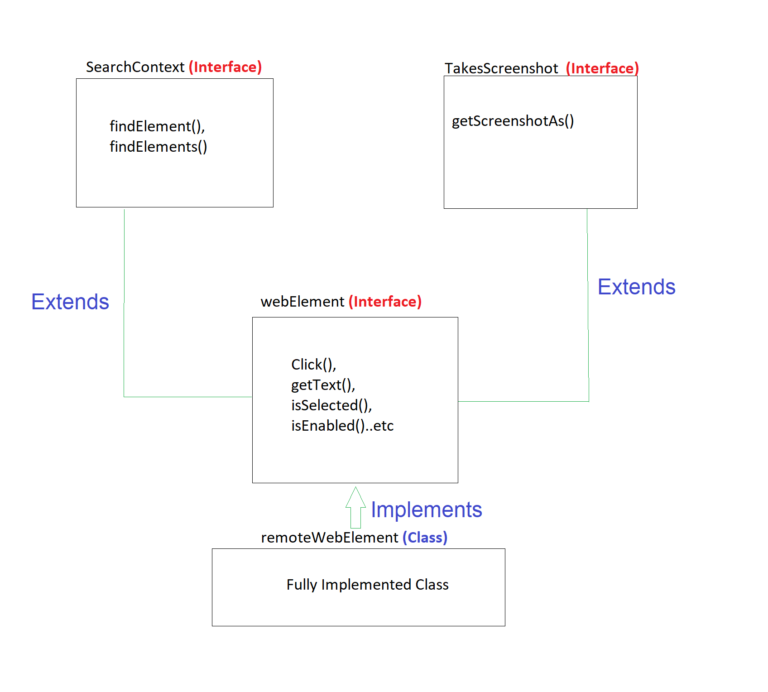
<class name=”com.TestNGExampleTests” />

</classes>

</test>

</suite>





**How to build a Framework in Selenium?**

* A test automation framework is a comprehensive set of guidelines and rules created to make test automation efficient and effective.
* A framework is comprised of a combination of test tools and practices that are designed to help QA specialist test more efficiently.
* Test Automation Frameworks is an essential for any successful automated testing process.
* A framework allows testers with less experience to manage sophisticated testing on shorter development cycles.

Frameworks incorporate components such as

* Coding standards
* Test data handling
* Object repository treatments
* Log configurations
* Rules for writing test cases
* Test results and reporting practices

**Data Driven Framework**

Strategy to access Excel Data

1. Create object for XSSF Workbook class
2. Get access to Sheet
3. Get access to all the rows of Sheet
4. Access to specific row from all rows
5. Get access to all cells of row
6. Access the data from Excel into Arrays

<http://www.pavantestingtools.com/>

<https://pavanonlinetrainings.com/index.html>

**How to work with Git & Github using Eclipse | Commit | Push**

<https://www.youtube.com/watch?v=HCeBd5GKNO8>

**Maven & Jenkins Integration with Selenium: Complete Tutorial**

<https://www.guru99.com/maven-jenkins-with-selenium-complete-tutorial.html>

**Github Integration with Selenium: Complete Tutorial**

<https://www.guru99.com/selenium-github.html>

Selenium Continuous Integration with Jenkins using GIT

<https://www.softwaretestingmaterial.com/selenium-continuous-integration/>

1. Selenium Tutorial by Simplilearn

<https://www.youtube.com/watch?v=Jdkrj2lDAEY&list=PLEiEAq2VkUUJALG6nbw0sY-zbhAAx5IuT&index=8>

2. Selenium Tutorial by Simplilearn

<https://www.youtube.com/watch?v=cobEbkTwbwY>

3. Git and Gihub <https://www.youtube.com/watch?v=xuB1Id2Wxak&list=PL9ooVrP1hQOE5ZDJJsnEXZ2upwK7aTYiX&index=12>

4. Jenkins Pipeline Tutorial

<https://www.youtube.com/watch?v=_fmX31VFbL8&t=19s>

5. Selenium Code snippet

<https://www.fromdev.com/2013/09/webdriver-selenium-code-snippets.html>

6. Selenium Titbits

<http://executeautomation.com/blog/selenium-titbits/?page-img41729=1>

7. Selenium Tutorial

<https://www.tutorialandexample.com/selenium-tutorial/>

8. Automation Framework in Selenium > <http://testautomationarchives.blogspot.com/2013/09/automation-frameworks-with-example.html>

Some practice web sites are as follows

<https://rahulshettyacademy.com/AutomationPractice/>

<https://admin-demo.nopcommerce.com/admin>

<http://demo.automationtesting.in/FileDownload.html>

<http://testingmasters.com/student-corner/downloads>

**Selenium Notes were taken from udemy by Rahul Shetty – rahulonlinetutor@gmail.com**

<https://hexaware.udemy.com/course/selenium-real-time-examplesinterview-questions/learn/lecture/15970090#overview>

<https://rahulshettyacademy.com/> & <http://qaclickacademy.com/>

<https://rahulshettyacademy.com/AutomationPractice/>

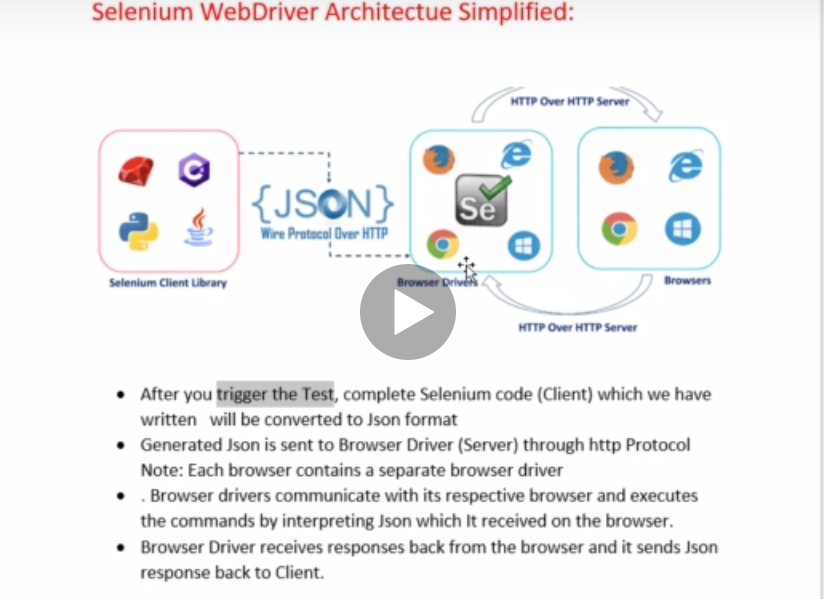
<https://rahulshettyacademy.com/angularpractice/>

<https://rahulshettyacademy.com/seleniumPractise/#/>

Top 70 Selenium Interview Question and Answers – Rahul Shetty

<https://www.youtube.com/watch?v=Hy7w-ls1cbc>

<https://dzone.com/articles/building-a-data-driven-keyword-driven-and-hybrid-s>

****

**Remarks**

1. Every object may not have ID, Name or ClassName. In that case XPath and CSS is preferred.
2. Alphanumeric id may vary on every refresh. Check
3. Confirm the link object with anchor “a” tag
4. Classes should not have spaces. Compound classes cannot be accepted.
5. Selenium identifies the first one scans from the top left if it contain multiple values
6. Double quotes inside double quotes are not accepted.
7. XPath/CSS can be defined in number of ways. Right click copy on blue highlighted html code to generate XPath. XPath allows to traverse forward from parent element to child element as well traverse back from child element to parent element.
8. Firepath plug-in is deprecated from Firefox browser version 55.
9. When XPath starts with /html tag, it is not reliable. Then switch to another browser [Chrome] to get the correct XPath.
10. There is no direct way to get CSS in chrome. You will find it in tool bar.
11. To validate the XPath and CSS from the browser add-ons

Right Click Browser > Inspect> Console > Enter $("") for CSS and $x("") for XPath

1. XPath Syntax : //tagName[@attribute=’value’]

XPath Syntax for Regular Expression: //tagName[contains(@attribute,’value’)]

1. CSS Syntax: tagName[attribute=’value’] or tagName#id & tagName.className

CSS Syntax for Regular Expression: tagName[attribute\*=’value’]

If id is present in the web page use #. If class is present then use dot[.] if there is a space between class attribute value then replace the space with the dot.

E.g. input identifyinput can be written as .input.identifyinput

**Selenium Interview Questions**

**Beginner Level**

1. What are the Selenium suite components?

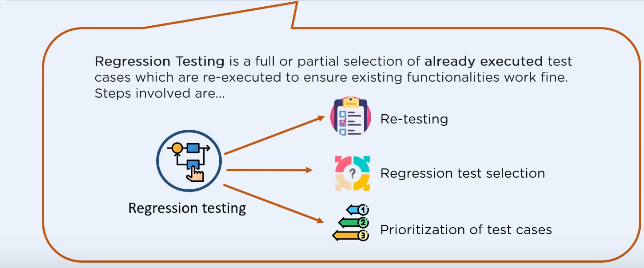
* Selenium IDE
* It is a Firefox/Chrome plug-in that was developed to speed up the creation of automation scripts.
* It records the user actions on the web browser and exports them as reusable script.
* Selenium RC
* RC is a server that allows users to write application tests in various programming languages.
* The commands from the test scripts are accepted by this server and are sent to the browser as Selenium Core JavaScript commands.
* The browser then behaves accordingly.
* Selenium Web Driver
* Web Driver is a programming interface that helps create and run test cases.
* It makes provision to act on web elements.
* Unlike RC, Web Driver does not require an additional server and interacts natively with the browser applications.
* Selenium Grid
* Grid was designed to distribute commands different machines simultaneously.
* It allows the parallel execution of tests on different browsers and different operating systems.
* It is exceptionally flexible and is integrate with other suite of components for simultaneous execution.

1. What are the limitations of Selenium?

* No reliable tech support.
* Tests only web applications.
* Does not handle windows GUI or not html popups - Selenium can be integrated with other tools like AutoIT or Robot class.
* Cannot automate Mobile application – Selenium can be integrated with Appium.
* Limited support for image testing – Sikuli can be integrated with Selenium.
* No built in reporting capability.
* Selenium does not integrate with any test management tool.
* May require knowledge of programming languages.

1. What are the testing types supported by Selenium?

Selenium supports Regression testing and Functional testing.



**Re-testing**

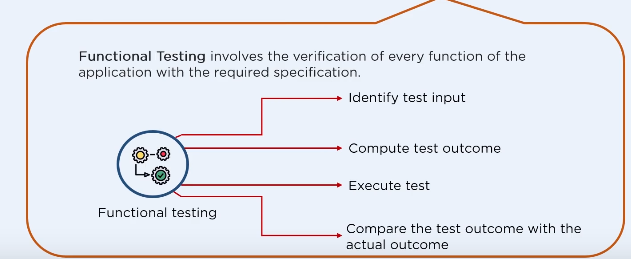
* All tests in the existing test suite are executed.
* Very expensive and time consuming.

**Regression testing -** Tests are classified as

* Feature tests
* Integration tests
* End to End tests
* Some of these tests are selected.

**Test Case Prioritization**

The selected test cases are prioritized based on the business impact and critical functionalities.

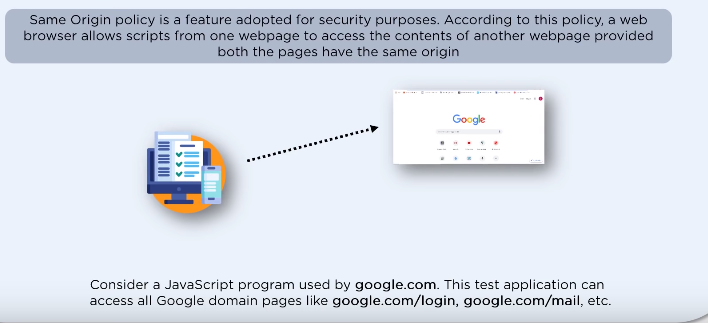


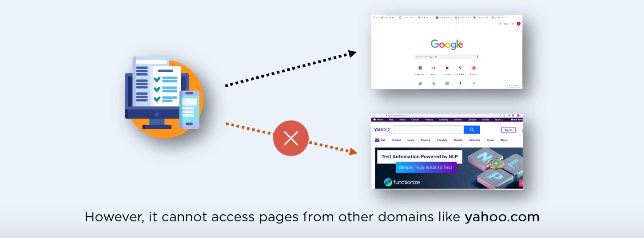
1. What are Selenium 2.0 and Selenium 3.0?

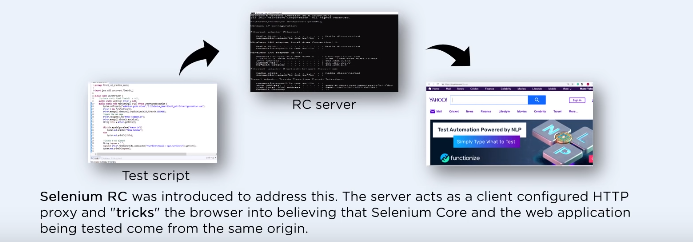


1. What is Same Origin Policy? How it is handled?

Origin is a combination of URL schema, hostname and port number. Same Origin Policy prevents a malicious script on one page to access sensitive data on another web page.

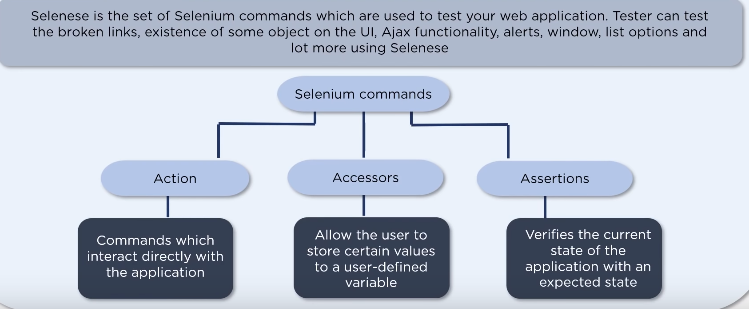






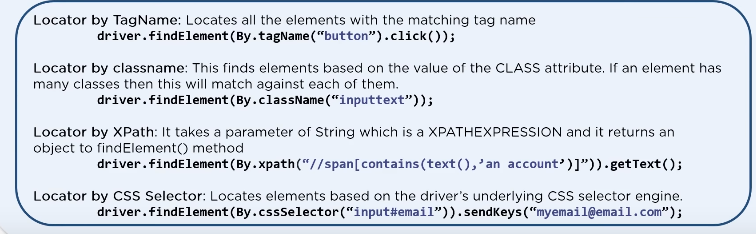
1. What is Selenese? How it is classified?

Selenium commands for Selenium IDE is called as Selenese.



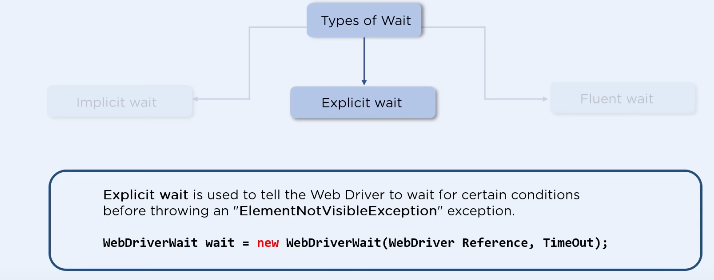
1. Mention the types of web locators?

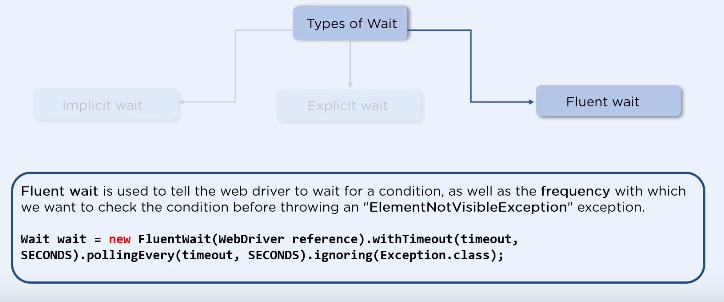




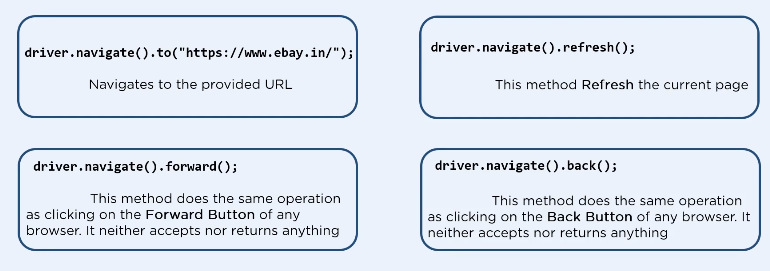
1. What are the types of waits supported by Web Driver?



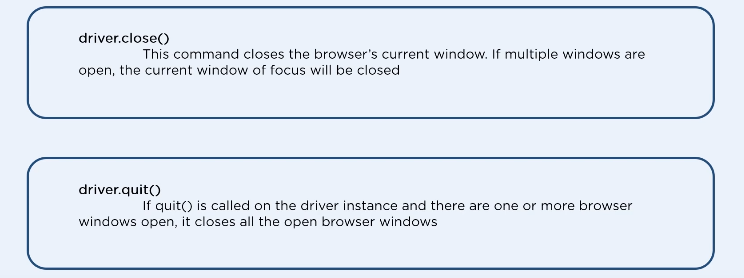




1. Mention the types of navigation commands?

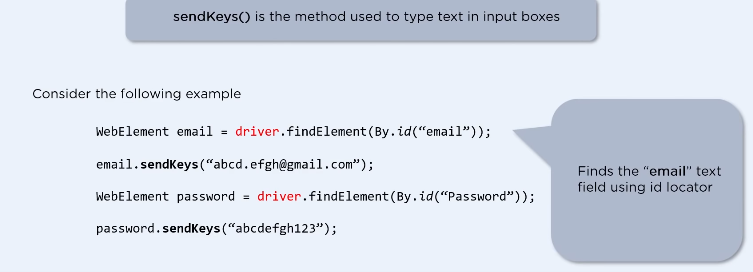


1. What is the major difference between driver.close() and driver.quit()?

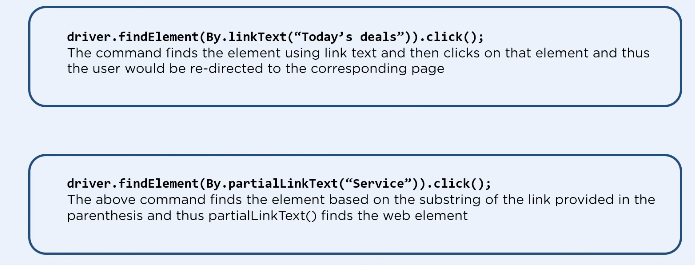


**Intermediate Level**

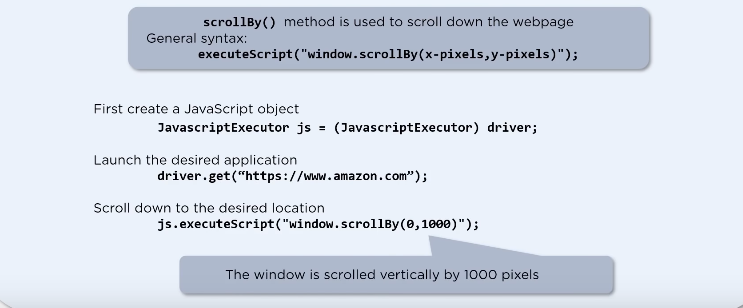
1. How to type text in an input box using Selenium?



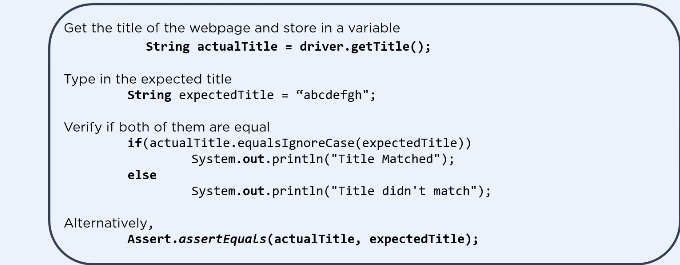
1. How to click on a hyperlink in Selenium?



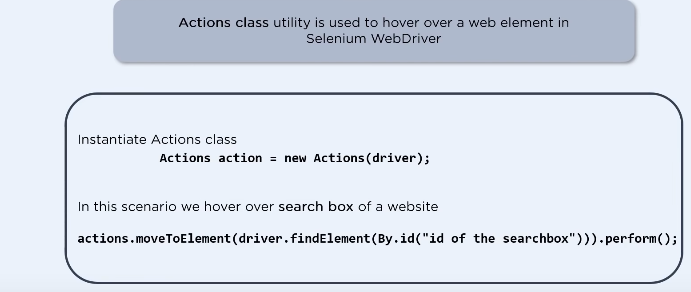
1. How to scroll down a page using JavaScript?



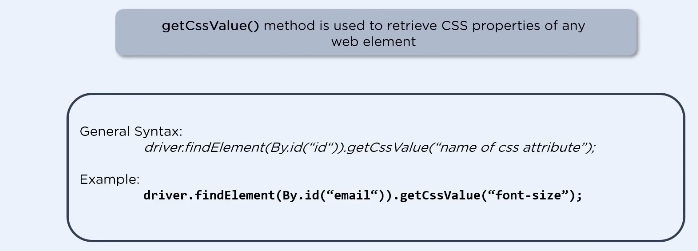
1. How to assert the title of a web page?



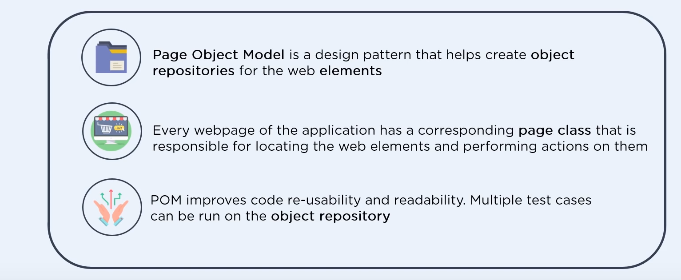
1. How to mouse hover a web element?



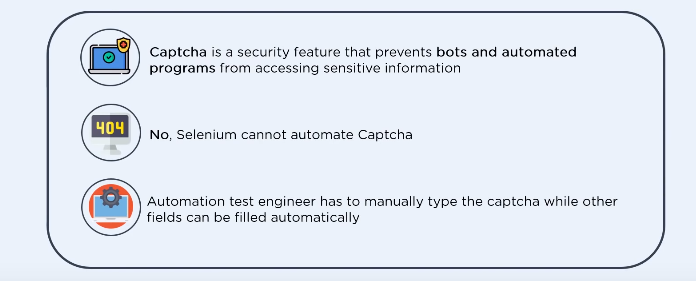
1. How to retrieve the CSS properties of an element?



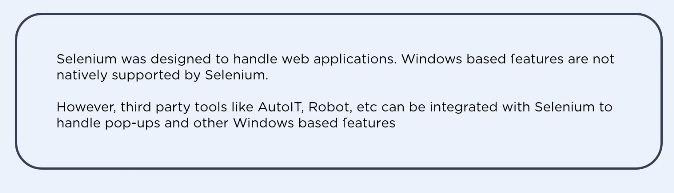
1. What is POM (Page Object Model)?



1. Can Captcha be automated?



1. How does Selenium handle Windows bases pop-up?

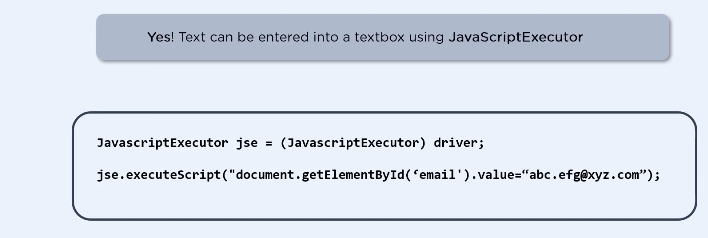


1. How to take screenshots in Selenium?

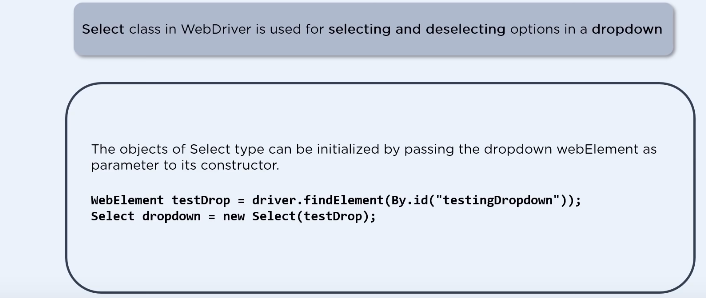


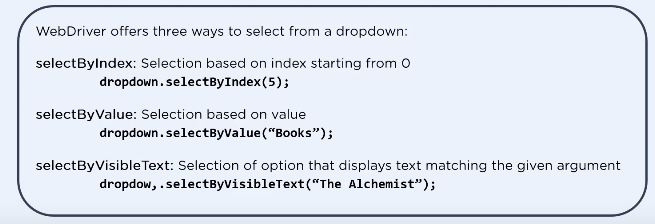
**Advance Level**

1. Can you type in a text box without using sendKeys()?

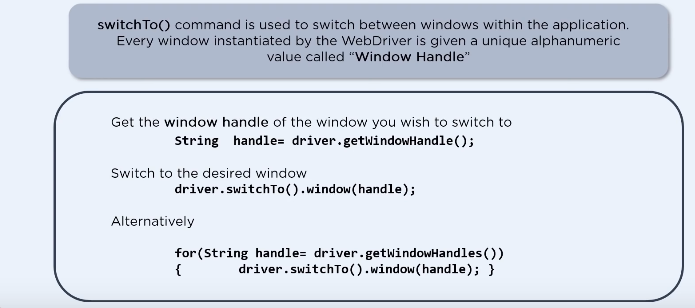


1. How to select a value from a dropdown in Selenium Web Driver?



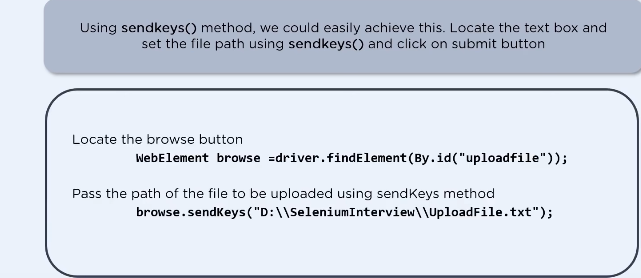


1. What does the switchTo() command do?

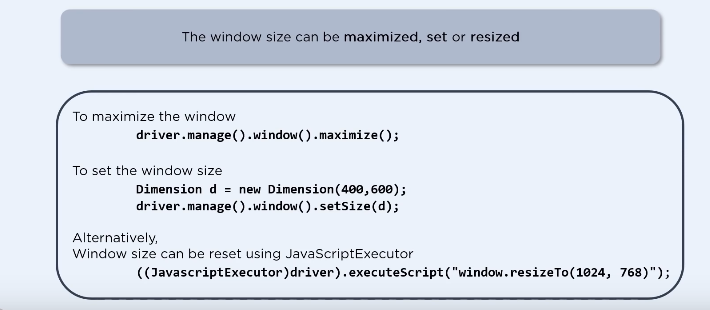


1. How to upload a file in Selenium Web Driver?

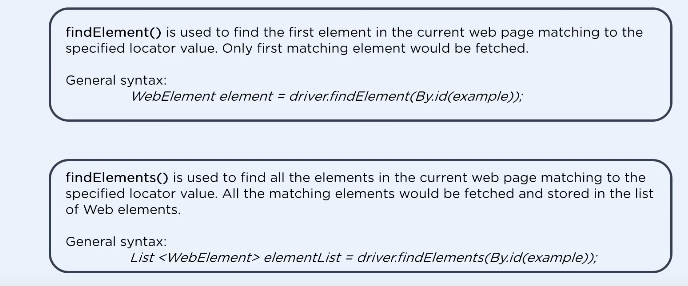
Note: tag should be <input> and attribute should be type=”File” then only you can upload a file.



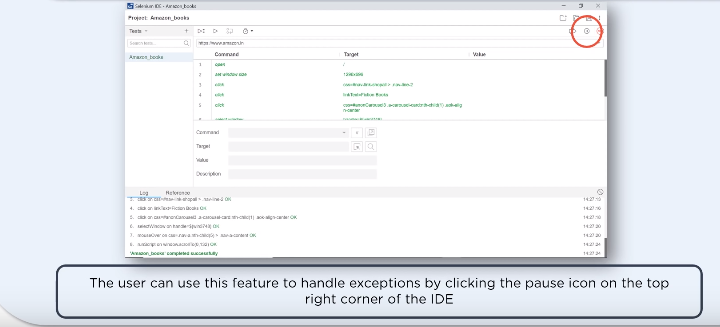
1. How to set bowser window size in Selenium Web Driver?



1. When do we use findElement() and findElements()?

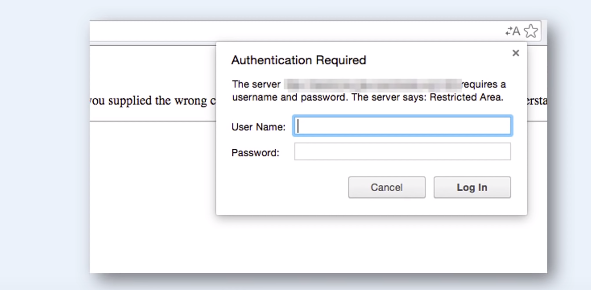


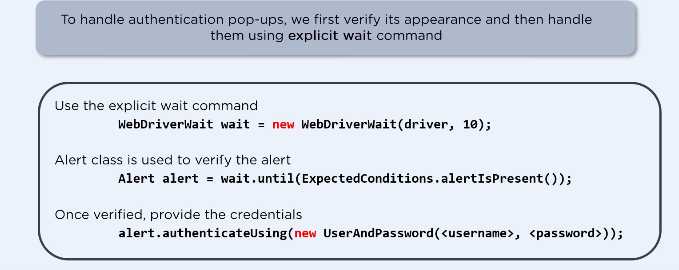
1. What is pause on exception in Selenium IDE?



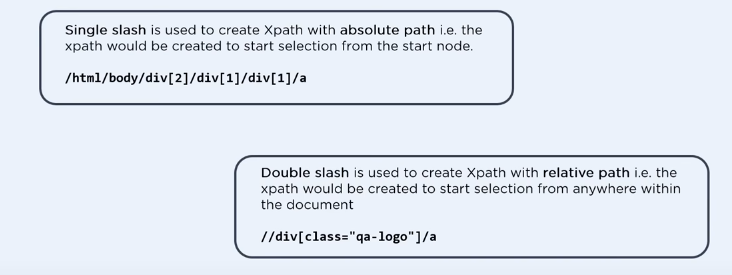


1. How to login to any site if it is showing an Authentication Pop-Up for Username and Password?

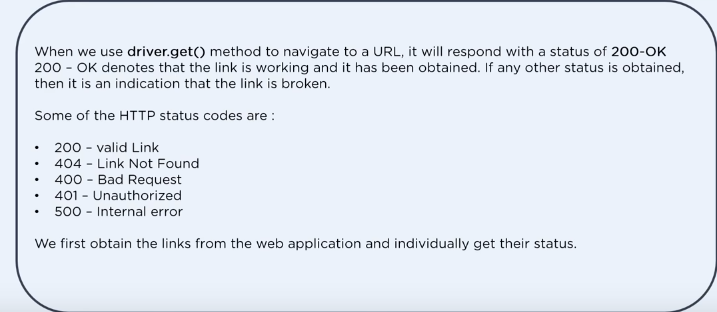




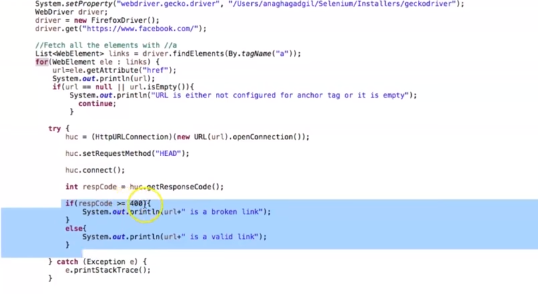
1. What is the difference between single and double slash Xpath?



1. How to find broken links in Selenium Web Driver?







**Interview Q&A**

1. How to traverse to sibling element using XPath?

.//\*[@id=’tablist1-tab1’]/following-sibling::li[2]

1. How to traverse back to parent element from the child element using XPath?

.//\*[@id=’tablist1-tab1’]/parent::ul

1. What is the difference between Relative and Absolute XPath?

Absolute xpath depends on the parent nodes, while relative xpath expression is independent of parent nodes.

Relative – Does not depend on Parent Nodes

Absolute – Parent/Child node

1. Identifying Text Objects using XPath

//\*[text()=’<string>‘]

1. Before writing the scripts in any IDE do the following
2. Verify in Firebug
3. Use CSS locator. i.e., #ID or .ClassName
4. Find the unique attribute
5. When will you get NoSuchElementException : Unable to locate element?

If the XPath is wrong you will get this exception.

1. When will you get ElementNotVisibleException : element not visible?

In any Ticket Booking System, if the city name is present in both From and To destination, and If the XPath is not the index based you will get this exception. i.e., XPath may occur more than once in the web page.

Other way is use Parent-Child Relationship XPath to identify uniquely

//div[@id='glsctl00\_mainContent\_ddl\_destinationStation1\_CTNR'] //a[@value='MAA']

1. Static dropdown – Select class || Dynamic dropdown with indexes

Parent Child relationship XPath & Auto suggestive dropdowns

**Sample Selenium scripts**

**Handling Dynamic dropdowns – Parent-Child relationship**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

public class Dropdown {

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

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

WebDriver driver =new ChromeDriver();

driver.get("http://spicejet.com"); //URL in the browser

driver.findElement(By.id("ctl00\_mainContent\_ddl\_originStation1\_CTXT")).click();

driver.findElement(By.xpath("//a[@value='BLR']")).click();

Thread.sleep(2000);

//driver.findElement(By.xpath("(//a[@value='MAA'])[2]")).click();

driver.findElement(By.xpath("//div[@id='glsctl00\_mainContent\_ddl\_destinationStation1\_CTNR'] //a[@value='MAA']")).click();

driver.findElement(By.cssSelector(".ui-state-default.ui-state-highlight.ui-state-active")).click();

}

}

**Parent and Following-Sibling**

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class testit {

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

System.setProperty("webdriver.chrome.driver","C:\\work\\chromedriver.exe");

WebDriver driver =new ChromeDriver();

driver.get("http://www.qaclickacademy.com/interview.php");

driver.findElement(By.xpath("//li[text()=' Selenium ']")).click();

driver.findElement(By.xpath("//ul[@class='responsive-tabs\_\_list']/li[1]/following-sibling::li[2]")).click();

System.out.println(driver.findElement(By.xpath(".//\*[@id='tablist1-tab2']/parent::ul")).getAttribute("role"));

}

}

**Assertions**

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

WebDriver driver =new ChromeDriver();

driver.get("http://spicejet.com"); //URL in the browser

Assert.assertFalse(driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).isSelected());

//Assert.assertFalse(true);System.out.println(driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).isSelected());

driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).click();

System.out.println(driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).isSelected());

Assert.assertTrue(driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).isSelected());

**Validating if UI elements are disabled or enabled with Attributes**

// System.out.println(driver.findElement(By.name("ctl00$mainContent$view\_date2")).isEnabled());

System.out.println(driver.findElement(By.id("Div1")).getAttribute("style"));

driver.findElement(By.id("ctl00\_mainContent\_rbtnl\_Trip\_1")).click();

System.out.println(driver.findElement(By.id("Div1")).getAttribute("style"));

if(driver.findElement(By.id("Div1")).getAttribute("style").contains("1"))

{

System.out.println("its enabled");

Assert.assertTrue(true);

}

else

{

Assert.assertTrue(false);

}

**Script using isDisplayed() method**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class formMethods {

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

WebDriver driver=new FirefoxDriver();

//Is displayed is used when particular object is in code base but it is in visible mode or not

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

System.out.println(" Before clikcing on Multi city Radio button");

System.out.println(driver.findElement(By.xpath(".//\*[@id='return\_date\_sec']")).isDisplayed());

//driver.findElement(By.xpath(".//\*[@id='multi\_city\_button']/span")).click();

//driver.findElement(By.xpath(".//\*[@id='multi\_city\_button']/span")).isEnabled();

System.out.println(" After clikcing on Multi city Radio button");

driver.findElement(By.xpath(".//\*[@id='start\_date\_sec']/span[3]")).click();

driver.findElement(By.xpath(".//\*[@id='ui-datepicker-div']/div[2]/table/tbody/tr[5]/td[3]/a")).click();

int i=0;

while(i<5)

{

driver.findElement(By.xpath(".//\*[@id='adult\_count']/a[2]")).click();

i++;

}

//System.out.println(driver.findElement(By.xpath(".//\*[@id='return\_date\_sec']")).isDisplayed());

//System.out.println(driver.findElement(By.xpath(".//\*[@id='mui\_city\_button']/span")).isDisplayed());

Thread.sleep(3000L);

//System.out.println(driver.findElement(By.xpath(".//\*[@id='responsive\_bottom']/div[2]/div[1]/div/div/h3")).getText());

//If you want to validate the object which is present in web page or code base

int count=driver.findElements(By.xpath(".//\*[@id='mui\_city\_button']/span")).size();

if (count==0)

{

System.out.println("verified");

}

}

}

**Flight booking E2E script**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

public class e2e {

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

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

WebDriver driver =new ChromeDriver();

driver.get("http://spicejet.com"); //URL in the browser

driver.findElement(By.id("ctl00\_mainContent\_rbtnl\_Trip\_0")).click();

driver.findElement(By.id("ctl00\_mainContent\_ddl\_originStation1\_CTXT")).click();

driver.findElement(By.xpath("//a[@value='DEL']")).click();

Thread.sleep(2000);

driver.findElement(By.xpath("//div[@id='glsctl00\_mainContent\_ddl\_destinationStation1\_CTNR'] //a[@value='MAA']")).click();

driver.findElement(By.cssSelector(".ui-state-default.ui-state-highlight.ui-state-active")).click();

if(driver.findElement(By.id("Div1")).getAttribute("style").contains("0.5"))

{

System.out.println("its disabled");

Assert.assertTrue(true);

}

else

{

Assert.assertTrue(false);

}

driver.findElement(By.cssSelector("input[id\*='SeniorCitizenDiscount']")).click();

driver.findElement(By.id("divpaxinfo")).click();

Thread.sleep(2000L);

for(int i=1;i<5;i++)

{

driver.findElement(By.id("hrefIncAdt")).click();

}

driver.findElement(By.id("btnclosepaxoption")).click();

Assert.assertEquals(driver.findElement(By.id("divpaxinfo")).getText(), "5 Adult");

System.out.println(driver.findElement(By.id("divpaxinfo")).getText());

// driver.findElement(By.cssSelector("#ctl00\_mainContent\_btn\_FindFlights")).click();

driver.findElement(By.cssSelector("input[value='Search']")).click();

// driver.findElement(By.xpath("//input[@value='Search']")).click();

// driver.findElement(By.name("ctl00$mainContent$btn\_FindFlights")).click();

}

}

# Find out broken links on website using selenium webdriver and HTTP Client

import java.util.List;

import org.apache.http.HttpResponse;

import org.apache.http.client.HttpClient;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.impl.client.HttpClientBuilder;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class FindBrokenLinksExample {

private WebDriver driver;

private int invalidLinksCount;

@BeforeClass

public void setUp() {

driver = new FirefoxDriver();

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

}

@Test

public void validateInvalidLinks() {

try {

invalidLinksCount = 0;

List<WebElement> anchorTagsList = driver.findElements(By

.tagName("a"));

System.out.println("Total no. of links are "

+ anchorTagsList.size());

for (WebElement anchorTagElement : anchorTagsList) {

if (anchorTagElement != null) {

String url = anchorTagElement.getAttribute("href");

if (url != null && !url.contains("javascript")) {

verifyURLStatus(url);

} else {

invalidLinksCount++;

}

}

}

System.out.println("Total no. of invalid links are "

+ invalidLinksCount);

} catch (Exception e) {

e.printStackTrace();

System.out.println(e.getMessage());

}

}

@AfterClass

public void tearDown() {

if (driver != null)

driver.quit();

}

public void verifyURLStatus(String URL) {

HttpClient client = HttpClientBuilder.create().build();

HttpGet request = new HttpGet(URL);

try {

HttpResponse response = client.execute(request);

// verifying response code and The HttpStatus should be 200 if not,

// increment invalid link count

////We can also check for 404 status code like response.getStatusLine().getStatusCode() == 404

if (response.getStatusLine().getStatusCode() != 200)

invalidLinksCount++;

} catch (Exception e) {

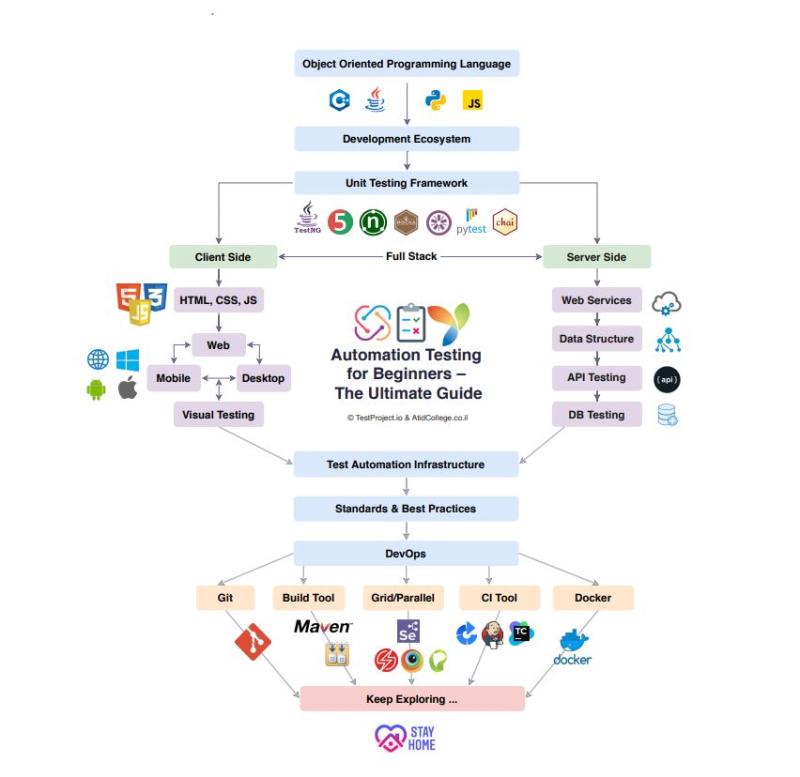
e.printStackTrace();

}

}

}

**Step-by-Step automation testing guide**



Please find the some of the excellent Selenium Urls given below

​<http://www.fromdev.com/2013/09/webdriver-selenium-code-snippets.html> > 15+ Useful Selenium Web driver Code Snippets For Web App Testing Automation

<http://executeautomation.com/blog/selenium-code-snippets-2/> > Selenium Code Snippets

<http://executeautomation.com/blog/selenium-titbits/?page-img41729=1> > Selenium Titbits

<http://executeautomation.com/blog/functional-tools/selenium-automation-with-c/> > Execute Automation By Karthik

<https://www.youtube.com/watch?v=TUX7ntXlVkc&list=PL6tu16kXT9PpteusHGISu_lHcV6MbBtA6&index=1> > Cucumber with Selenium by Karthik Execute Automation

<http://viralpatel.net/blogs/20-useful-java-code-snippets-for-java-developers/> > 20 very useful Java code snippets for Java Developers

<http://www.fromdev.com/2008/05/java-collections-questions.html> > 24 Java Collections Framework Interview Questions & Answers for developers

<https://cucumber.io/docs> > Cucumber Documentation

<https://www.youtube.com/watch?v=yPtnml4tNII>  > selenium framework from scratch

<https://www.youtube.com/watch?v=_7XJenTvR34> > Data Driven Framework

<https://www.youtube.com/watch?v=dvqjmj70ltw> > Selenium Jenkins Tutorial ANT Maven GIT GITHUB Integration

# 20 Fundamentals for a Successful Test Automation Infrastructure

**1.**[Identify Page Elements](http://blog.testproject.io/2017/02/09/inspect-web-elements-chrome-devtools/) (text fields, buttons, link etc.) and configure them in a certain class, in which you can implement functions that will perform several operations. For example: inserting text and clicking on a button by identifying the element beforehand – which is called [Page Object](http://blog.testproject.io/2017/02/09/page-object-pattern-advantages-implementation/).

**2.** Write a mechanism that will be activated when a test fails.

**3.** Write another mechanism to return the page to a certain point (for example: the main page) in order to create independent tests.

**4.** [Develop a report module](http://blog.testproject.io/2017/02/09/develope-reporting-module-test-automation-framework/) and[implement a report module](http://blog.testproject.io/2017/02/09/implement-report-module-test-automation-framework/) for the executed automated tests.

**5.** Add to the reports screenshots and recordings when tests fail.

**6.** Implement the project in a way that will allow reading all parameters from external configuration files (for example: the location to save the reports).

**7.** Incorporate Visual Testing for image comparison to test if the site looks as it is supposed to (for example: if a field moved by 6 pixels to the right, the test will fail).

**8.**[Create Cross-browser compatibility.](http://blog.testproject.io/2017/02/09/cross-browser-testing-selenium-webdriver/)

**9.** Write log files to save information for the developers which will also include stack trace and other details from the exceptions, which will show up in case of a failed test.

**10**. Object Repository – implement also a central place in which you’ll identify all of the elements’ properties in the application.

**11.** Connect the system’s database to execute queries and draw information (Despite not having access to the application’s database, we can easily create a simulation for a portfolio).

**12.** Implement the Data Driven Testing (DDT) method to[read from CSV/ Excel files](http://blog.testproject.io/2017/02/09/read-data-csv-file-in-c/).

**13.** Implement the [Keyword Driven Testing (KDT)](http://blog.testproject.io/2015/09/15/kdt-keywords-driven-testing-vs-test-automation-scripts/) or BDD methods with Gherkin language.

**14**. Create support for parallel testing on different environments.

**15.** Connection to external libraries (for example: interface cloud services API such as YouTube)

**16.** Create support for scheduling execution and interface CI systems (it’s possible to install [Jenkins CI](https://blog.testproject.io/2017/05/11/jenkins-ci/) server on the computer and working with it).

**17.** Compatibility with Mobile device platforms.

**18.** Expand to a compatibility with [Desktop Apps](http://blog.testproject.io/2016/12/22/open-source-test-automation-tools-for-desktop-applications/).

**19.** The test automation infrastructure should be written according to known Design Patterns, Clean Code methods, coding conventions that must be configured at the design level.

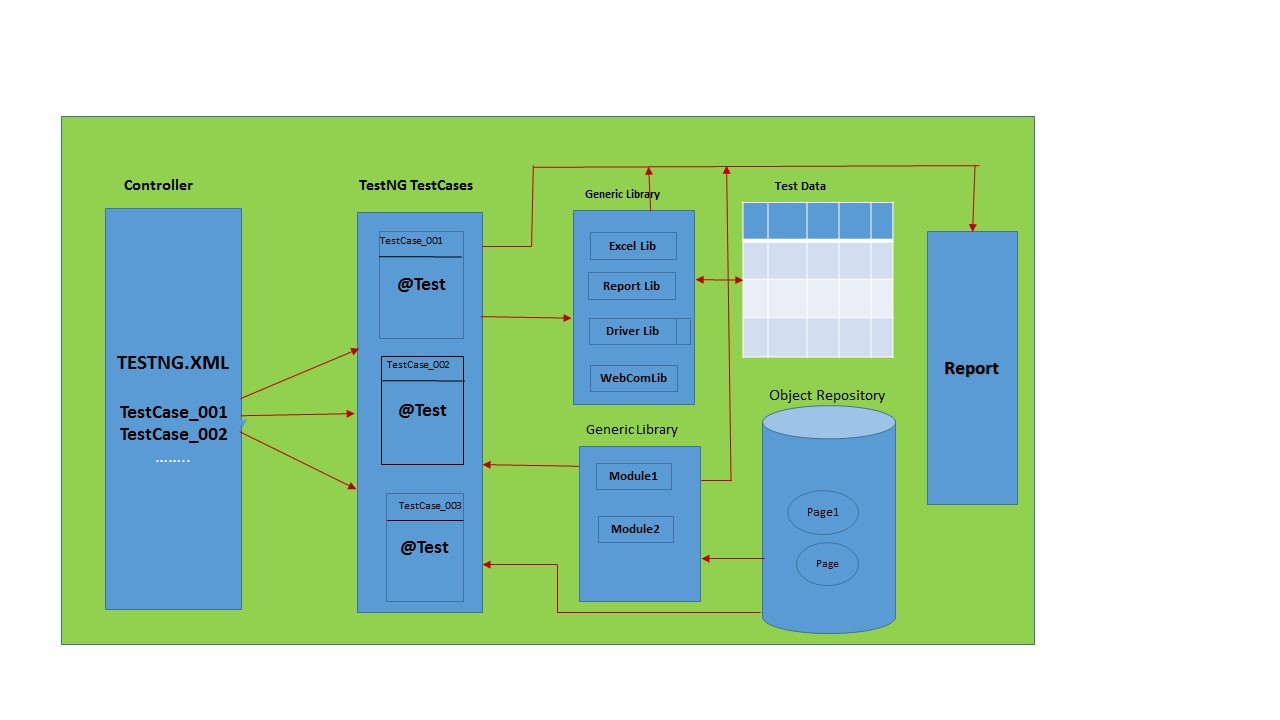
**20.** Finally, documentation of the entire procedure from A to Z.  
The list clearly shows how [Selenium](http://blog.testproject.io/2016/10/16/better-selenium-tests-webdriverio/) is a very small component of the automation testing project.

<https://betsol.com/2017/08/how-to-get-started-selenium-automation-framework/>

**Key benefits of adopting an automation framework**

A framework defines the organization’s way of doing things. A test automation framework provides a standard for simplifying the automation effort. The end results are a reduction in software testing costs and maintenance costs.

* The ability to test large volumes of data on the system quickly.
* Scripting standard will be maintained across the framework in library creation, which includes business components, system communications, data check points, loggers, reporters, and more.
* Engineering benefits including increased code re-usage, higher portability, and reduced script maintenance cost, higher code readability, and more.
* Licensing costs are reduced through mature open-source solutions, such as Selenium Web Driver.
* Automated regression test suites that massively reduce manual effort while also increasing quality.



### Selenium automation framework components

**TestNG** is a testing framework inspired from **JUnit** and **NUnit,** but introduces new functionality that makes it more powerful and easier to use. TestNG test case are built using TestNG annotations. Each testing class can have multiple Test case, but test cases are distributed on the packages basis on application modules

Test Data is driven through a testData Excel sheet. Each row of data is dedicated to one test case.

Generic libraries contains multiple Java files that support various features of the framework, like customized reporting, test data fetching, and web driver-specific method. Any application independent reusable method, which can be used for any projects such as Excel\_Lib, Reporting\_Lib, WebdriverCommonUtil, and Driver Java class can be classified under Generic Libraries.

For example, You may create a “Driver” class as shown below to create the choice of browser based on the requirement.

public class BDriver {

public static WebDriver driver = new FirefoxDriver();

public static WebDriver IEDriver = new InternetExplorerDriver();

public static WebDriver cromeDriver = new ChromeDriver();

}

You can create “Excel\_Lib” class containing few reusable methods that can fetch test data from Excel sheet.

Similarly, you can implement WebdriverCommonUtil class that contains web driver specific custom method that can be used for multiple applications such as clickAndWait(), waitForPageToLoad(), waitForElementPresent(), and more.

**Logging using Log4J API**

For any web application automation, the web events are helpful to view the events triggered by Web driver. Whenever you develop test scripts, you should write your own implementation for handling events during the execution.

When you run test cases, you may want to implement logging to see when the test started executing, when the test failed and what is the message / warning it has provided.

Log4j is built with component that work together to enable logging of messages according to the message type and level, and to control at runtime how these messages are formatted and where they are reported.

### Controller framework – TestNG XML

The [TestNG XML](http://testng.org/doc/documentation-main.html#testng-xml) file is the brain of our automation framework, which takes care of executing the testing classes as specified in the XML file. With this XML file, you may plan and control of the automated test cases execution:

* Declare environment variable (global variable)
* Prioritize and execute test cases in order
* Execute batch execution and grouping parallel execution

Also you can skip particular method or particular test in test suite

This framework is driven through TestNg.xml file, which has details of the test suiteA test suite contains a bunch of testNg classes and each tesNg classes can have multiple test cases. For example, a sample of testng.xml is shown here:

<?xml version=”1.0″ encoding=”UTF-8″?>

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

<suite name=”Automated Test suite” >

<test name=”Regression Test suite”>

<classes>

<class name=”TestClassName1″ />

<class name=”TestClassName2″ />

<class name=”TestClassName3″ />

…

</classes>

</test>

</suite>

<https://www.testingexcellence.com/develop-test-automation-framework-scratch/>

When creating a Test Automation Framework, we should consider the following main points:

To be able to create automated tests quickly by using appropriate abstraction layers

The framework should have meaningful logging and reporting structure

Should be easily maintainable and extendable

Should be simple enough for testers to write automated tests

A retry mechanism to rerun failed tests – this is especially useful for WebDriver UI tests

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.

Why Framework?

In a test automation project, we do perform different tasks by using different types of files. To organize and manage all the files and to finish all the tasks in a systematic approach we use a framework.

**Selenium Interview Scenarios**

<https://www.scientecheasy.com/2019/12/testng-interview-questions.html>

**How can you add the screenshot for the only failed scenario in a test?**

* We will write 2 test cases (scripts/ methods) in a single test class (FirstTest.java).
* First Test script would be to open the browser, navigate to <https://www.google.com/>, enter some text and click on Google Search button.
* The second test would only contain one line of code *Assert.fail(),*to fail the test case deliberately.

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.AfterSuite;

import org.testng.annotations.BeforeSuite;

public class BaseClass {

public static WebDriver driver;

//Browser should only opened once during whole test execution

@BeforeSuite

public void setUp() {

System.setProperty("webdriver.chrome.driver", "/home/dhawal/Downloads/chromedriver/chromedriver");

driver = new ChromeDriver();

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

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

}

//Browser will be closed after all the methods in FirstTest are finished execution

@AfterSuite

public void tearDown() {

driver.quit();

}

}

import org.openqa.selenium.By;

import org.testng.Assert;

import org.testng.annotations.Test;

public class FirstTest extends BaseClass{

//Enter some text and click on Search button

@Test

public void search() {

driver.findElement(By.name("q")).sendKeys("Applied Selenium");

driver.findElement(By.xpath("//div[@class='FPdoLc VlcLAe']//input[@value='Google Search']")).click();

}

//Deliberately fail the test case

@Test

public void feelingLucky() {

Assert.fail();

}

}

import java.io.File;

import java.io.IOException;

import java.util.Calendar;

import java.util.GregorianCalendar;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.io.FileHandler;

public class TestUtil extends BaseClass {

public static String screenshotpath;

public static void captureScreenshot(String methodName) throws IOException {

//Using GregorianCalendar to fetch the time details

Calendar cal = new GregorianCalendar();

//Month value is always 1 less than actual. For February, MONTH would return 1

int month = cal.get(Calendar.MONTH);

int year = cal.get(Calendar.YEAR);

int sec = cal.get(Calendar.SECOND);

int min = cal.get(Calendar.MINUTE);

int date = cal.get(Calendar.DATE);

int day = cal.get(Calendar.HOUR\_OF\_DAY);

//Take screen shot and store it in volatile memory with reference name scrFile

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

try {

//Create unique file name and store in under screenshot folder of root directory

screenshotpath = System.getProperty("user.dir") + "/screenshot/" + methodName + "\_" + year + "\_" + date+ "\_" + (month + 1) + "\_" + day + "\_" + min + "\_" + sec + ".jpeg";

//Copy the file to destination

FileHandler.copy(scrFile, new File(screenshotpath));

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

import java.io.IOException;

import org.testng.ITestContext;

import org.testng.ITestListener;

import org.testng.ITestResult;

public class ScreenShotListener extends BaseClass implements ITestListener {

@Override

public void onTestStart(ITestResult result) {

// TODO Auto-generated method stub

}

@Override

public void onTestSuccess(ITestResult result) {

// TODO Auto-generated method stub

}

//Override below method to implement screenshot

@Override

public void onTestFailure(ITestResult result) {

try {

TestUtil.captureScreenshot(result.getName().toString());

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

@Override

public void onTestSkipped(ITestResult result) {

// TODO Auto-generated method stub

}

@Override

public void onTestFailedButWithinSuccessPercentage(ITestResult result) {

// TODO Auto-generated method stub

}

@Override

public void onStart(ITestContext context) {

// TODO Auto-generated method stub

}

@Override

public void onFinish(ITestContext context) {

// TODO Auto-generated method stub

}

}

**In a test case there are 10-12 assertion if one of them gets failed, rest of the assertion is not getting executed. How can we solve the problem?**

A **soft assert** will run the test and not throw an exception if the assert failed, while a **hard assert** will throw the exception immediately, and then continue with the testing process.

**There are few Test Ng test case , how can we add the dependency within the methods?**

To run the test cases in a specific order, then we use the concept of dependency in TestNG. Two types of dependency attributes used in TestNG

**dependsOnMethods**  
The dependsOnMethods attribute tells the TestNG on which methods this test will be dependent on, so that those methods will be executed before this test method.

import org.testng.annotations.Test;

public class Login

{

@Test

public void login()

{

System.out.println("Login page");

}

@Test(dependsOnMethods="login")

public void home()

{

System.out.println("Home page");

}

}

**dependsOnGroups**

Similar to the dependsOnMethods attribute. It allows the test methods to depend on the group of test methods. It executes the group of test methods before the dependent test method

import org.testng.annotations.Test;

public class Test\_cases

{

@Test(groups="smoke")

public void testcase1()

{

System.out.println("testcase1");

}

@Test(groups="smoke")

public void testcase2()

{

System.out.println("testcase2");

}

@Test(dependsOnGroups="smoke")

public void testcase3()

{

System.out.println("testcase3");

}

}

**How can we handle the dynamic Web Table using selenium?**

**Get Maximum of all the Values in a Column of Dynamic Table**

import java.text.ParseException;

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import java.text.NumberFormat;

public class MaxFromTable {

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

WebDriver wd;

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

wd= new ChromeDriver();

wd.get("http://demo.guru99.com/test/web-table-element.php");

String max;

double m=0,r=0;

//No. of Columns

List col = wd.findElements(By.xpath(".//\*[@id='leftcontainer']/table/thead/tr/th"));

System.out.println("Total No of columns are : " +col.size());

//No.of rows

List rows = wd.findElements(By.xpath (".//\*[@id='leftcontainer']/table/tbody/tr/td[1]"));

System.out.println("Total No of rows are : " + rows.size());

for (int i =1;i<rows.size();i++)

{

max= wd.findElement(By.xpath("html/body/div[1]/div[5]/table/tbody/tr[" + (i+1)+

"]/td[4]")).getText();

NumberFormat f =NumberFormat.getNumberInstance();

Number num = f.parse(max);

max = num.toString();

m = Double.parseDouble(max);

if(m>r)

{

r=m;

}

}

System.out.println("Maximum current price is : "+ r);

}

}

**Page Object Pattern:**

1. Easy to maintain
2. Script readability is easy
3. Eliminate duplicate codes
4. Code reusability
5. Reliability

**What are listeners?**

1. **Listeners are basically the ones who have the ability to listen to a particular event.**
2. **It is defined as an interface that modifies the behavior of the system.**
3. **Listeners allow customization of reports and logs.**

**Types of listeners in Selenium**

1. **WebDriver listeners**
2. **TestNG listeners**

**WebDriver listeners**

**WebDriverEventListener interface allows to implement methods and classes like WebDriverEventListener and EventFiringWebDriver.**

**WebDriverEventListener**

1. **This is an interface which holds some predefined methods. These Web driver events are helpful to view the events triggered by the web driver.**
2. **It plays an important role in analyzing the results and helps us in debugging issues if we encounter any.**
3. **It has ability to track different events like “beforeNavigateTo” , “afterNavigateTo”, “BeforeClickOn”, “AfterClickOn” and many more.**

**Steps to capture a screenshot in selenium**

1. **Create a class. Implement TestNG ‘ITestListener‘.**
2. **Call the method ‘onTestFailure’.**
3. **Add the code to take a screenshot with this method.**
4. **Get the Test method name and take a screenshot with the test name. Then place it in the desired destination folder.**

**Override the following methods in ITestListener Interfce**

onTestStart, onTestSuccess, onTestFailure, onTestSkipped, onTestFailedButWithinSuccessPercentage >>> For all these methods ITestResult is the parameter

result.getName();

onStart, onFinish >>> for these 2 methods ITestContext is the parameter.

context.getOutputDirectory();

context.getPassedTests();

context.getFailedTests();

**Create common utility function to capture the screenshot**

**public** **class** CommonUtil {

**public** **static** **void** captureScreenshot(WebDriver driver, String screenshotName)

{

**try** {

TakesScreenshot scrnShot = (TakesScreenshot)driver;

File source = scrnShot.getScreenshotAs(OutputType.***FILE***);

FileUtils.*copyFile*(source, **new** File("src\\test\\resources\\"+screenshotName+".png"));

System.***out***.println("Screenshot taken .... ");

} **catch** (WebDriverException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.getMessage();

}

}

}

/\*

//@AfterMethod

public void tearDown(ITestResult result)

{

if(ITestResult.FAILURE==result.getStatus())

{

Utility.catureScreenshot(driver,screenshotName);

}

driver.quit();

}

\*/

**Add he Listeners annotation in the respective class**

@Listeners(com.demo.example.ListenersTest.class)

**Make an entry in testng.xml**

<listeners>

<listener class-name=*"com.demo.example.ListenersTest"* />

</listeners>

**Note: RandomStringUtils.randomAlphabetics(5);**

**Desired Capabilities in Selenium**

* Capabilities: Describes a series of key/value pairs that encapsulate aspects of a browser.
* Desired Capabilities is a class used to declare a set of basic requirements such as combinations of browsers, operating systems, browser versions, etc. to perform automated cross-browser testing of a web application.
* A typical usecase would be to set the path for the FirefoxDriver if your local installation doesn't correspond to the default settings.
* It is a class in org.openqa.selenium.remote.DesiredCapabilities package.
* It gives facility to set the properties of browser. Such as to set BrowserName, Platform, Version of Browser.
* Mostly DesiredCapabilities class used when do we used Selenium Grid.
* We have to execute mutiple TestCases on multiple Systems with different browser with Different version and Different Operating System.

**How does Maven SureFire Plugin works?**

* Maven sure fire plugin is used to follow the sequence of tests in testng. xml file.
* If we don't include the Mavwen surefire plugin then it will execute all the testcases under src/test/java which has prefix or suffix as 'test' and these tests will get executed without any sequence.