**What testing you performed in using Selenium?**

Smoke-Major functionality like-Login, Logout etc.

end to end- scratch to end.

Regression – check the older functionality in new application/software.

Selenium can be integrated with multiple tools such as TestNG, AutoIt, Maven, Jenkins and so on.

**Selenium IDE**-It gives you flexibility to convert your code into multiple languages and use it accordingly.

Some plugin that we can use with IDE is screenshot on failure, highlighter, export code into excel and so on.

But Selenium IDE has some of the limitation like-

* It is only available in Firefox so we can record your script in Firefox only.
* Selenium IDE does not have good reporting feature which generally we use for reporting to managers and lead or Team.
* Selenium IDE does not support parallel execution which is one of the most important features of Automation.
* Selenium IDE does not support remote execution as well.

**Selenium RC**- is not another tool or plugin it is just library which contains several packages classes interfaces and methods.

Selenium RC had so many features like-

* It supports almost all browser which is available in the market.
* We can easily integrate Selenium RC with TestNG which makes Selenium more powerful.
* Selenium RC having very good reporting feature with the help of TestNG.
* Now we can do execution in parallel which the help of TestNG. We can pass multiple parameters, data driven and all the features which TestNG have.
* Selenium RC came with Remote execution which we can achieve through Selenium grid.

**Selenium Grid 1.0**-Selenium Grid has node and hub concept which works in Client-Server architecture concept. We can create one central hub which can connect multiple Node. Node can be different machine like Windows, Linux, Mac, Android, IOS and so on.

**Limitations of Selenium RC-1**- It generally interact with Server it means every time you run script it will send request to server then server will communicate with browsers.

Performance was an issue here. Since 2010 Mobile industry is leading in market so Selenium RC was not capable to perform.

Note- Selenium RC officially deprecated now in market.

**Selenium Webdriver- Selenium 2**

Selenium Webdriver came in 2011 and It supports all the feature of Selenium RC and additionally they have following benefits.

* They removed server part from it so performance not an issue in Webdriver. It means simply write your code and it will directly communicate with browsers.
* Selenium Webdriver supports Mobile Automation as well which make Selenium more powerfull. Currently Selendroid and Appium are present which allow us to automate IOS and Android application.

**Pain of an automation tester-**

When I used to create automation script

* My script used to work fine sometimes and sometimes it used to fail.
* When I run the script in my local machine it works fine but when I run in remote or another browser it fails.
* Scripts work fine in one browser and it fails in other browsers.
* Scripts work fine for current release, but it fails when new release come for my applications.
* Earlier I used to create scripts without any framework so even if small changes in the application then I used to make the changes in each script.

**Best practice to overcome this pain.**

* Start writing your own xpath using xpath methods.
* Start using a design/Framework-You can directly start writing your automation script without any framework that fine but maintenance task will be a big headache once test case size will grow even if a small change in the application then you have to modify all script.
* You can start using any framework, but it should have below characteristic. -

1. Reusable 2- Easy to use 3- Easy to maintain 4- Robust in nature

* Execute your script multiple times -It may take some time but your script will be highly stable and you will get good ROI from your script.
* Use Smart wait in your script-90 % automation script failure reason is Sync issue and locator changes only.

**IE browser** is very much concerned about security you will find a couple of Issues with IE browser

If you are using Selenium 3 then to work with Firefox browser, you need to use separate driver which will interact with Firefox browser.

**How to execute Selenium Webdriver in Chrome Browser?**

Your test case will fail and you will get IllegalStateException which says we need to specify the chrome driver path where it resides. If you notice Selenium also gives a very meaningful message that we need to add some chrome variable also while running the script.

**Variable name is – webdriver.chrome.driver**

In Java to set variable we use setProperty method of System class so let us add the same in our program.

**System.setProperty("webdriver.chrome.driver", "path of the exe file\\chromedriver.exe");**

# How to remove Disable Developer Mode Extension in Selenium?

// Create object of ChromeOptions class

ChromeOptions options = new ChromeOptions();

// add parameter which will disable the extension

options.addArguments("--disable-extensions");

// Start the chrome session

WebDriver driver = new ChromeDriver(options);

**Some key point while working with IE Browser in Selenium.**

* IE browser is slow as compared to other browsers.
* Your browser zooming level should be set to 100 % otherwise, you will get an exception.
* You have to check your security setting also in IE. While running IE browser in Selenium your all zones should be either enabled or disabled. If not, then again you will get an exception and your test cases will fail.

## **Challenges with IE browser in Selenium Webdriver**

You will get the different type of Exception while working with IE Browser

**Issues 1-**

### openqa.selenium.NoSuchWindowException

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

**Issue 2-**

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

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

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

**Issue 3-**

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

When I started working with IE this was the first exception, which I used to get, and I was sure that this related to some browser setting.

**Issue 4-**

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

By the name itself, you can see that we have to set the zoom level to 100 % to make it work.

**Issue 5:**

**Handle Untrusted SSL certificate error in IE browser in different ways  
Solution: IE is the product of Microsoft and IE is much worried about security so when you start working with some https application you will get a untrusted certificate.**

**Selenium has so many ways to handle this, but we will see 2 ways which work all the time for me.**

**First:**Open the application for which SSL certificate is coming so use below code after passing the URL.  
driver.get(“ur app URL”);  
driver.navigate().to(“javascript:document.getElementById(‘overridelink’).click()”);  
// you can use your code now

**Second:**

You can handle this certificate using Desired Capabilities as well.

# How to Automate Radio button and Checkbox in Selenium webdriver

Before performing click action, sometimes we need to verify some activity as well, take some example

* You need to verify whether radio button or checkbox is enabled.
* You need to verify whether radio button or checkbox is Displayed on UI or not.
* You need to verify whether checkbox and radio button is default selected or not.

Above validations are must use in script because automation is all about validation only.

we can easily [***verify***](http://learn-automation.com/capture-error-message-in-selenium/)this using some predefined method in Selenium.

|  |
| --- |
| isDisplayed();  isEnabled();  isSelected(); |

WebElement male\_radio\_button=driver.findElement(By.id("u\_0\_e"));

**boolean** status=male\_radio\_button.isDisplayed();

System.out.println("Male radio button is Displayed >>"+status);

**boolean** enabled\_status=male\_radio\_button.isEnabled();

System.out.println("Male radio button is Enabled >>"+enabled\_status);

**boolean** selected\_status=male\_radio\_button.isSelected();

System.out.println("Male radio button is selected >>"+enabled\_status);

# How to Handle Dropdown in Selenium WebDriver

WebElement month\_dropdown=driver.findElement(By.id("month"));

Select month=**new** Select(month\_dropdown);

month.selectByIndex(4);

WebElement month\_dropdown=driver.findElement(By.id("month"));

Select month=**new** Select(month\_dropdown);

month.selectByValue(“5”);

WebElement month\_dropdown=driver.findElement(By.id("month"));

Select month=**new** Select(month\_dropdown);

month.selectByVisibleText("Aug");

**Get Selected option from Dropdown.**

WebElement month\_dropdown=driver.findElement(By.id("month"));

Select month=**new** Select(month\_dropdown);

WebElement first\_value=month.getFirstSelectedOption();

String value=first\_value.getText()

**Get All option from dropdown**

WebElement month\_dropdown=driver.findElement(By.id("month"));

Select month=**new** Select(month\_dropdown);

List<WebElement> dropdown=month.getOptions();

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

String drop\_down\_values=dropdown.get(i).getText();

System.***out***.println("dropdown values are "+drop\_down\_values);

}

Explanation- getOptions() is a method of Select class which will return List of WebElement then we can iterate using for loop or iterator and using getText() method we can extract values.

getAllOptions().size()-1 which will always select last value from Dropdown.

# How to Handle Bootstrap Dropdown in Selenium WebDriver

**First one – What is bootstrap dropdown**

**The second one- How to Select values from the bootstrap dropdown.**

The bootstrap dropdown is enhanced part of dropdown where you will deal with UL and LI tag of HTML.To handle this kind of drop-down we have to use findElements method and then we can run a for loop to get specific elements.

// Dropdown items are coming in <a> tag so below xpath will get all

// elements and findElements will return list of WebElements

List<WebElement> list = driver.findElementsByXPath("//ul[@class='dropdown-menu']//li/a");

// We are using enhanced for loop to get the elements

**for**(WebElement ele:list)

{

// for every elements it will print the name using innerHTML

System.***out***.println("Values " + ele.getAttribute("innerHTML"));

// Here we will verify if link (item) is equal to Java Script

**if** (ele.getAttribute("innerHTML").contains("JavaScript")) {

ele.click();

// break the loop or come out of loop

**break**;

}

}

// here you can write rest piece of code

}

You can also select the values directly using [xpath](http://learn-automation.com/write-dynamic-css-selector-in-selenium/)  and [CSS](http://learn-automation.com/write-dynamic-css-selector-in-selenium/) but that approach is not recommended because direct xpath might change.

In above approach, we can pass a parameter directly so based on test data it will select the values from the list.

# Upload file in Selenium webdriver using Robot class

**There are multiple ways to upload files in Selenium Webdriver**

* You can directly use sendKeys(“File path”) method but sometime this methods does not work.
* We can use Robot class to upload files in Selenium.
* ***We can upload files using AutoIT as well.***

While working with robot class I faced so many issues like it works on Latest window only

***Robot class is a separate class in Java which will allow us to perform multiple tasks based on our requirement. It generally will throw AWT exception so we need to deal with it accordingly.***

Using Robot class, we can simulate keyboard event in Selenium.

To use keyboard events you have to use to a method of Robot class.

## Robot Class in Selenium Webdriver

keyPress()

keyRelease()

Each key has to be press and release respectively-

// Create object of Robot class

  Robot r=new Robot();

   // Press Enter

  r.keyPress(KeyEvent.VK\_ENTER);

   // Release Enter

  r.keyRelease(KeyEvent.VK\_ENTER);

**Actions Class-**

For all advance activity in Selenium Webdriver, we can perform easily using Actions class like Drag and Drop, mouse hover, right click, Click and Hold and so on.  
We have predefined method called **dragAndDrop(source, destination)** which is a method of Actions class.

Approach- Find the xpath of the Source  and find the xpath of destination.

Both source and destination in form of WebElement.

Note- Any method of Actions class we need to call perform () method otherwise we will get anexception. If we have series of action in our script using Actions class then we have to call build().perform() method.

// Create object of actions class

Actions act=new Actions(driver);

// find element which we need to drag

WebElement drag=driver.findElement(By.xpath(".//\*[@id='draggable']"));

// find element which we need to drop

WebElement drop=driver.findElement(By.xpath(".//\*[@id='droppable']"));

// this will drag element to destination

act.dragAndDrop(drag, drop).build().perform();

We can also perform drag and drop using x and y coordinate but this will be only applicable when destination is not given.

moveToElement(WebElement)-- Mouse Hover

contextClick()-- Right click on page

contextClick(WebElement)-- Right click on specific Element

sendKeys(KEYS.TAB)--For keyboard events

clickAndHold(WebElement)--Click on element and hold until next operation

release() Release the current control

Actions act=new Actions(driver);

act.contextClick(driver.findElement(By.linkText(“Gujarati”))).perform();

Actions act=new Actions(driver);

 act.contextClick(driver.findElement(By.linkText("Gujarati"))).sendKeys(Keys.ARROW\_DOWN).sendKeys(Keys.ARROW\_DOWN).sendKeys(Keys.ENTER).build().perform();

**// Verify error message**

Assert.assertEquals(actual\_msg, expect);

Note- If text does not match then TestNG will throw AssertionError and if we do not use Exception handling then it will simply terminate our application/ program.

# Handle Multiple Windows in Selenium Webdriver

If you have to switch between tabs then also you have to use the same approach.

In Selenium, we have the feature that we can get the window name of the current window.In Selenium, we have the getWindowName method that will return current window name in String form.

We also have getWindowNames, which will return Set<String> it means the set of window name then we can iterate using Iterator. The set is part of Java collection which allows us to handle multiple sets of data dynamically.

**How to handle alert in Selenium Webdriver**

Web-Based alert and Java Script alerts are same so do not get confused.

How to Handle alert pop up in selenium Webdriver

To handle alert window in Selenium Webdriver we have predefined Interface known as Alert .

1- accept()- Will click on the ok button when an alert comes.

2- dismiss()- Will click on cancel button when an alert comes.

Note– Since alert is separate window so before using these methods we have to switch to alert window using switchTo() method

Now consider a scenario where alert window comes when certain condition true for this we can create method which will check if alert window present then only it will execute otherwise it will skip this part

public static void handleAlert(WebDriver ldriver){

if(isAlertPresent(ldriver)){

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

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

alert.accept();

}

}

public static boolean isAlertPresent(WebDriver ldriver){

try{

ldriver.switchTo().alert();

return true;

}catch(NoAlertPresentException ex){

return false;

}

}

Important point- If alert in not present in the window and still we try to switchTo alert window then Selenium will throw NoAlertPresentException which will terminate your program so better you should use exception handle also in your script.

## **Programs to handle frames in selenium**

**Syntax 1-**

In this scenario, if you know the total number of frames in the web page then using the index, you can easily switch.

The index generally starts with zero so if you have only one frame then the index will be zero. If you don’t know the total number of frames in the page then you can use findElementBytagname method.

try {

driver.switchTo().frame(indexnumber);

      }

catch (NoSuchFrameException e)

     {

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

      }

We have enclosed our code with try and catch if now frame will not available this throw exception NoSuchFrameException

**Syntax 2-**

In this scenario, if you know the name  of frames in web page then using name also, you can easily switch

 try {

driver.switchTo().frame(“framename”);

      }

catch (NoSuchFrameException e)

    {

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

    }

**Syntax 3-**

In this scenario

try {

WebElement button=driver.findElement(By.xpath(""));

driver.switchTo().frame(button);

}

catch (NoSuchFrameException e)

     {

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

     }

**Until you are in frames you can not perform any operation so once we are don with frame then switchTo parent window**

|  |  |
| --- | --- |
| driver.switchTo().defaultContent(); |  |

**How to capture Screenshot in Selenium webdriver**

For taking  screenshots Selenium has provided TakesScreenShot interface in this interface you can use getScreenshotAs method which will capture the entire screenshot in form of file then using FileUtils we can copy screenshots from one location to another location

// Take screenshot and store as a file format

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

try {

 // now copy the  screenshot to desired location using copyFile //method

FileUtils.copyFile(src, new File("C:/selenium/error.png"));

}

catch (IOException e)

 {

  System.out.println(e.getMessage());

 }

Now consider a scenario where you have to take multiple screenshots then above code will be repetitive so for this we will create a small method which captures screenshots and will call this method from our script.

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

**public** **static** **void** captureScreenShot(WebDriver ldriver) {

// Take screenshot and store as a file format

File src = ((TakesScreenshot) ldriver).getScreenshotAs(OutputType.FILE);

**try** {

// now copy the screenshot to desired location using copyFile method

FileUtils.copyFile(src, **new** File("C:/selenium/" + System.*currentTimeMillis*() + ".png"));

}

**catch** (IOException e)

{

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

}

}

# How to capture screenshot for failed test cases in Selenium Webdriver

### **Generally, scripts fail in 2 situations.**

1-If script has some issue (some locator has been changed or application has some changes)- In this case, we need to maintain our Selenium script.

2-Due to application issue- In this case, we need to inform to respective point of contact (Manual Tester or Developer)

1-We will use ITestResult Interface which will provide us the test case execution status and test case name.

2- @AfterMethod is another annotation of TestNG which will execute after every test execution whether test case pass or fail @AfterMethod will always execute.

// It will execute after every test execution

@AfterMethod

public void tearDown(ITestResult result)

{

// Here will compare if test is failing then only it will enter into if condition

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

{

try

{

// Create refernce of TakesScreenshot

TakesScreenshot ts=(TakesScreenshot)driver;

// Call method to capture screenshot

File source=ts.getScreenshotAs(OutputType.FILE);

// Copy files to specific location here it will save all screenshot in our project home directory and

// result.getName() will return name of test case so that screenshot name will be same

FileUtils.copyFile(source, new File("./Screenshots/"+result.getName()+".png"));

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

}

catch (Exception e)

{

System.out.println("Exception while taking screenshot "+e.getMessage());

}

## **Limitation of screenshots by Selenium**.

1- When any alert comes on screen and if you call screenshot method then it will fail because the alert is windows activity.

If you are not aware of Alert in Selenium and how to handle, then check out below article to get more info.

2- When running cross browser testing if need to verify that test is running on which browser then you won’t be able to verify because it captures only web view par

### Create highlight element method for reuse

public static void highLightElement(WebDriver driver, WebElement element)

{

JavascriptExecutor js=(JavascriptExecutor)driver;

js.executeScript("arguments[0].setAttribute('style', 'background: yellow; border: 2px solid red;');", element);

try

{

Thread.sleep(500);

}

catch (InterruptedException e) {

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

}

js.executeScript("arguments[0].setAttribute('style','border: solid 2px white');", element);

}

### **What is Base Class in Selenium**

* Base class in the main class which will take care of Browser setup, loading [**configuration file**](http://learn-automation.com/object-repository-in-selenium-webdriver/)and other reusable methods like [**screenshot**](http://learn-automation.com/how-to-capture-screenshot-for-failed-test-cases-in-selenium-webdriver/), handling [**sync issues**](http://learn-automation.com/best-way-to-handle-synchronization-in-selenium-webdriver/)and many more.
* Using Base class we can avoid code duplication.
* Reuse code as much we can.

### How Base class works in Selenium

1-When we create base class and if TestCases extends BaseClass then we can use all the methods of Baseclass.

2- Before calling actual @Test Base class methods will get executed and Depends on annotations it will call the respective methods.

3- We can extend this class in all test cases and we can call custom methods as well directly.

----------------------------------------------------------------------------------------------------------------------------

Selenium always opens new profile/fresh instance of browser by default along with no addons/extensions. So there is no need to clear history separately.

**is it not possible to set zooming level 100% and security settings in ie using selenium?**

DesiredCapabilities caps = DesiredCapabilities.internetExplorer();

caps.setCapability(“EnableNativeEvents”, false);

caps.setCapability(“ignoreZoomSetting”, true);

WebDriver driver = new InternetExplorerDriver(caps);

--------------------------------------------------------------------------------------------------------------------------------

DesiredCapabilities cap=DesiredCapabilities.chrome();

// Set ACCEPT\_SSL\_CERTS variable to true

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

System.setProperty("webdriver.chrome.driver","Chrome driver path");

WebDriver driver=new ChromeDriver(cap);

--------------------------------------------------------------------------------------------------------------------------------

// Create object of DesiredCapabilities class

DesiredCapabilities cap=DesiredCapabilities.internetExplorer();

// Set ACCEPT\_SSL\_CERTS  variable to true

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

System.setProperty("webdriver.ie.driver","IE driver path");

WebDriver driver=newInternetExplorerDriver(cap);

--------------------------------------------------------------------------------------------------------------------------------

//It create firefox profile

FirefoxProfile profile=new FirefoxProfile()

// This will set the true value

profile.setAcceptUntrustedCertificates(true);

// This will open firefox browser using above created profile

WebDriver driver=new FirefoxDriver(profile);

driver.get("pass the url as per your requirement");

text() function always looks for exact matching of text while contains(text(),”) doesn’t look for exact matching text.

**I have following scenario :**

**Click on + Add button then it will add new row in the table.Now here whatever new row added in the table then its hard to find xpath. Second thing i want to add multiple dynamic xpath and want to enter text.**

If you are able to locate + button using xpath then after click on button, try dynamic xpath like //button[text()=’+’]/following::tr[1] to get first row. In similar fashion, you can proceed with other rows. This is just an example to navigate. Actual xpath on your application might differs.

--------------------------------------------------------------------------------------------------------------------------------

Google verification and Capcha code cant be automated using Selenium.

--------------------------------------------------------------------------------------------------------------------------------

**Can we use combination of xpath and css?**

NO

--------------------------------------------------------------------------------------------------------------------------------

**Please check whether the locators are correct. Moreover in some applications, sendKeys for input box doesn’t work in one shot, so first do click inside input box and then try with sendKeys action.**

**When the dropdown tag element is not a ‘select’ but a ‘button’**

Selenium by default supports only select tag. If you are seeing other than select then you need to go for some workaround.

If we are using drop down but having div class instead of select class what needs to be done

In this case you can use findElements() method which will return list of elements then you can select the item.

--------------------------------------------------------------------------------------------------------------------------------

**If we have multiple dropdown and all dropdown is depend on previous one selection dropdown. How we will execute second one or how to write the script in webdriver.**

Ex:- one dropdrown for Mobile Brand name.Second dropdown for Mobile Model so mobile model name depend on Brand name. ?

generally use explicit wait to handle this scenario

-------------------------------------------------------------------------------------------------------------------------------

**I actually finding different scenario like drag from local machine and drop in website. such as go to and drag a object from machine and drop on specified area. is there any method available in action or other class?? without using any third party tool.**

Actions class comes from Selenium APIs’ and selenium is built for performing actions on webpage only. Therefore, usage of third party tool is must if you want to drag and drop an item from your machine to webpage.

Selenium works with browsers only so if you want to work with local drive then you have to check other tools like Sikuli and AutoIT.

**What is difference between build() and perform() method?**

build() is called when you have to compile more than one action in a single step or in simple words it like adding sequence of actions to buffer and later we use perform() to execute it.

If you have single operation then its ok but when you have multiple operation then build().perform() is must.

--------------------------------------------------------------------------------------------------------------------------------

**Default timeout is ZERO. Default polling time is 250 millisecond if implicit wait given.**

--------------------------------------------------------------------------------------------------------------------------------

My code works fine with implicit wait on firefox. When I run the same code in chrome, I get No such element exception. I increased the implicit wait time to 100 seconds but still it throws no such element exception as soon as it logs in before waiting for 100secs (implicit wait time). The only way I could get my code work in chrome is by adding explicit wait on each element. Can you please let me know why is implicit wait not working in chrome in my case? Is there any other solution than adding explicit wait to each element to all my scripts?

If you are working with cross browsers, then sometime xpath will change. Try to use CSS which will remain same for all browsers.

--------------------------------------------------------------------------------------------------------------------------------

I am just wandering how your code is working without using System.SetProperty(“webdriver.chrome.driver”, “chromedriver.exe path”).

Yes, we can do if you set the path in Env variable then You don’t have to define the path in every script.

--------------------------------------------------------------------------------------------------------------------------------

**I have below doubts:**

**1. If we do not use any types of wait statements in program, what is default time for which webdriver will search for an element?**

**2. Is there any timeout for web page loading in selenium?**

**3. Can we increase wait time at run time means can we make webdriver to wait for some extra time to find out en element?**

1- by default wait is zero and if some wait are given then by default polling is 250 mili second.

2- Yes page load timeout is present if you want to set page load timeout. If no timeout given then page will wait until full page loaded.

3- At run time we can’t change the time out.