**24-3**-**2023 -- Summary**

**Alerts: -**

It is the interface between the user interface and web current web page.

It is a small message box and ask permission for moving and reducing risk factors.

We can’t automate the alerts.

Whenever alert is present the web page is inactive.

**Types :-**

**Simple alerts**

**prompt alert**

**Confirmation alert**

**Simple alerts--🡪**It is the alert which gives some information**.**

**prompt alert🡪**these pop ups may require some data. It requires a function called Sendkeys.

**Confirmation alert🡪**It provides options like accept /ok and dismiss/cancel.

**Syntax**: -

Driver. switchTo (). Alert. Accept () -🡪 for accepting the alert.

Driver. switchTo (). Alert. Dismiss ()--🡪for dismissing the alert.

Driver. SwitchTo (). Alert. get Text ()-🡪 for getting the text**.**

**Pop-up’s :-**

Pop up’s are similar to alerts. But here a window is appeared on the screen.

**1.** Alert popup

2.Notification popup

3.Hidden division popup🡪we can’t inspect it.

4.file download popup

5.child browser popup

**Actions class: -**

Generally, mouse actions should perform manually. But in automation with the help of actions class we can perform some of the mouse actions only. Initially we need to create instance for action class.

Actions actions=new Actions(driver)

With the help of this ref variable, we can perform actions.

Some mouse actions are

1.move to element—for performing mouseover action

actions. moveToElement (). build (). Perform ()

2.click --- for performing click operation

actions. Click (). Build (). perform()

3.context click—for perform right click operation

actions. Context click (). Build (). perform ()

4.drag and drop-for performing the drag and drop operation

actions. drag And drop (). Build (). Perform ()

5.double click-for perform double click operation

actions. double click (). Build (). Perform ()

**Robot class: -**

This is the one type of class used to perform keyboard and other mouse actions.

Robot robot=new robot ()

With the help of this reference variable, we can perform keyboard actions.

Whenever we are pressing a key, we automatically release that key.

Ex-I write a code for copy.

robot. KeyPress (KeyEvent.VK\_ctrl)

robot. KeyPress (Key Event. VK\_c)

robot. KeyRelease (Key Event .VK\_c)

robot. Key Release (Key Event .VK\_ctrl)

whenever the page is in active state then only it works.

**Exception Handling in selenium: -**

Exception is nothing but an error which occurs in the execution of the program and it terminates the execution of the entire program.

There are 2 types of exceptions: -

1.Checked exception—appears in compile time

2.Un checked exceptions---appears in run time.

Differences b/w checked and unchecked exceptions.

|  |  |
| --- | --- |
| **Checked** | **Un checked** |
| It is appeared in compile time | These are appeared in run time |
| These must be handled while writing the code | These should be ignored/not visible in run time |
| These are compile time exception. | These are called run time exceptions. |

How to handle:-

We can handle the exceptions by using Try and catch block.

try {

Code---

} catch (Exception e) {

e. printstacktrace ()

}

If the exception is a known exception just simply **throws** the exception.

e. printstacktrace () :-used to print the exception in console.

Every time we need to write a catch block or final block along with try block.

Finally block--irrespective of exception it should executes.

**Exceptions in selenium: -**

**1.No such element exception**

**2.Element is not interceptable**

**3.Stale exception**

**4.Web driver exception**

**5.IO exception**

**6.Timeout exceptions**

**7.No such frame exception.**

These are some common exceptions in selenium.

**Listeners: -**

**It listens and monitor the each and every event that occurs in selenium.**

1.TestNG listeners

2.webdriver listeners.

**TEST NG: -**

It states that test next generation. It is a open source test framework. Which is inspired by J and N units.

Which is the upgraded form of j unit because it uses **annotations** and **prioritization,** and **detailed report** can be given by test ng that is how many testcases should be passed and failed and skipped.

We can executes no of testcases parallelly by using testng.xml file

Cross browser testing is possible.

Should give the priority to the test cases as (priority=0).

Debugging is easy while writing the test cases in test ng.

**TestNG Annotations**: -

Used to control the flow of methods. It plays a vital role in executing methods.

@Before suite—it is executed only once before all the tests have run

@Before Test—it is executed before the first @test method. It executes multiple times before Test case execution.

@Before class- it is executed only once before the test case execution

@Before Method-it executed before every @test annotated method.

@Test –executed all the methods under test

@After method- it executed after every @test annotated method.

@After class- it is executed only once after the test case execution.

@After Test-- it is executed after the first @test method. It executes multiple times before Test case execution.

@After suite-- it is executed only once after all the tests have run.

**Flow of execution**: -

<BeforeSuite>

<Before Test>

<Before Classes>

<Before Method>

<Test>

<After Method>

<After Classes>

<After Test>

<After Suite>

Whenever we are using **@Test** annotations to method that is treated as testcase.

We can prioritize the testcases with @Test(priority=0). Here highest priority give to 0,1,2,3 like that.

Most commonly used methods in TestNg are: -

Assert. Assert Equals ()

Assert. assert True ()

Assert. assert False ()