What is Selenium? Selenium is a free (open-source) automated testing framwork used to validate web applications across diffrent browsers and platforms. you can ~~youse~~ use multiple programming language like Java, C#, python etc To Create Selenium test Scripts.

Selenium is a Suite of tools which can automate actions on webbrowsers.

Day 1 Basic of Java

- Basic Syntax ✓
- Classes ✓
- Constructor ✓
- variables ✓
- object Identfiers
- loops
- Strings
- Basic operators ✓

JUNIT

- @TEST - anotation is used to run a junit tests.
- @Before - anotation is used to run any specific test before each test
- @BeforeClass: This method executes once before running all test
- @After: This method executes after each test
- @AfterClass: Like @BeforeClass, @Afterclass executes once after all tests
- @parameters: annotation tag is used to pass multiple data.

Junit Assertion

Junit assertEquals: This checks two values are equal and the assertion fails if both values are not equals. This compairs Boolean, ints, Strings, float, long, char etc.

Assert.assertEquals("excepted value", "actual value");

JUnit AssertTrue : Returns true if the condition is true and assertion fails if the condition is fails.

Assert.assertTrue ("message", condition);

Junit assertfails : Return true if the condition is false and assertion fails if the condition is true:

Assert.assertFalse ("message", condition);

Step 1. WebDriver driver;

in @ before

Step2 System.setproperty ("webdriver.gecko.driver", driverPath+ "geckodriver.exe");

//open browser

⇒ driver = new firfoxDriver();

// open AUT

⇒ driver.get(url);

Step3 in @Test

// fetch the page title

⇒ String pageTitle = driver.getTitle();

⇒ Syso (pageTitle);

Diffrent types of WebElements

- Edit box
- Link
- Button
- Image, Image Link, an Image button.
- Checkbox.
- Radio button
- Dropdown List.

Locators

- ID
- Name
- Classname
- LinkText
- PartialLinkTest
- CSSselector
- Xpath
- TagName.

There are two methods of find webElements.

1. findElement() - finds a single webElement
2. findElements() - Return the list of webElements.

⇒ operations for texbox / editbox.

1) Enter A value - SendKeys()
2) Clear the value - clear()

⇒ operations that can be performed on Button.

1) Click
2) Submit

⇒ Interacting with dropdowns. → Radio button, check box, Drop down List.

1) SelectByVisibleText() / deselectByVisibleText()
2) SelectByValue() / deselectByValue()
3) SelectByIndex() / deselectByIndex()
4) isMultiple()
5) deselectAll()

Ex

```
webElement Select_Dropdown = driver.findElement(By.Id("list_D"));
Select chose_Dropdown = new Select(Select_Drpdown);
chose_Dropdown.SelectByIndex(1);
```

Now Printing table data

⇒

```
// fetch the Account details table
WebElement T_A_Detail = driver.findElement(By.id("offersTable"))
// fetch all rows from table by "tr" tage and store in Row ele
List<webElement> rows = T_A_Details.findElements(By.tagName("tr
// Print the number of Rows.
    Syso(" rows.size());
// Iterate over all the rows.
    for( WebElement row : rows ){
        // find all the cols inside the particular row using ("td".
        List<WebElement> Cols = row.findElements(By.tagName("td
        // Iterate all the coloms.
        for (WebElement Col : Cols) {
            Syso("Column value" + Col.getText())
        }
    }
```

Intraction With Frames

frames Can switched using either.

1. // Index
2. // Id
3. // Name

using Index -: `driver.switchTo().fram(1);`

```
// Switch back to parant frame.
driver.switchTo.parentFrame();
```

Frame name -: `driver.switchTo().fram("Frame_Name");`

Window Handling

```
Set <String> windowHandles = driver.getWindowHandles();
    // Iterate through all the avaible windows.
for( String handle : windowHandles) {
    // Switch b/w Window using the String reference variable.
driver.switchTo().window(handle);
    // fatch the URL of the page post Successfull switch.
String URL = driver.getCurrentUrl();
Syso(URL);

// if(URL.  check wether the URL post swith is desired page
if( URL.equals("http://  ")) {
        break;
}
}
```

Handle Pop-up

* pop-up is call as an alert

→ **Simple Alert** -:

1. Click on the xpath();
2. String message = driver.switchTo().alert().getText()
   System.out.println(" message);
   // Accept the alert.
3. driver.switchTo().alert().accept(); // Accepted.

⇒ **Confirmation Alert (Accept or Reject)** -:

- → driver.switchTo.alert().accept() // For OK
- → driver.switchTo().alert().dismiss // For Cancel.

③ Prompt Alert (This Alert Ask Some input from the user):

- → driver.SwitchTo.alert().SendKey ("mary");
  → driver.SwitchTo.alert().accept();

Action: Drag and Drop, Mouse Hover

① In the web page we would have two element to first one dragable and IInd Dropable Alement to of drag and drop Action:

Exa

Step 1: // get element Source which needs to drag.

- WebElement Source = driver.findElement (By.Id ("draggable"));
  // get position where element will be drop.
- WebElement ~~Sour~~ target = driver.findElement (By.Id ("droppable"));

Step 2:

Actions action = new Actions (driver);

Step 3:

// using actions class object, provide Source and target as webElement.
// then use build and perform method to drag drop.

- action.dragAndDrop (Source, target).build().perform();

→ ② mouse Hover Action

// get xpath of

- webElement userName = driver.findElement (By.xpath ("---"));
  // get xpath of sub button
- webElement logout = driver.findElement (By.xpath ("---"));
  // Creat Action class.
- Actions action = new Actions (driver);
- action.moveToElement (userName).perform();
- action.moveToElement (logout).click().build().perform();

Explicit Wait

Explicit wait :- If we given 20 Sec of Explicit wait then that will execute the program until the execution done within 20 Second. if execution done in 1 Sec then that will perform next task.

Exa

```
WebDriverWait wait = new WebDriverWait(driver, 20);
wait.until(ExpectedConditions.visibilityOfAllElementsLocatedBy
        (By.id("lblmessage3")));
```

```
String title = driver.findElement(By.id("lblmessage3").getTe
```

Working with Excel files

Implementing Class	Description
HSSFWorkbook	High level Representation of a workbook in .xls file. we mostly use it to reading and writing a workbook. It is the top level object for Creating a new /sheets.
HSSFSheet	High level Representation of a workbook in .xls file.
HSSFRow	High level Representation of a row of a Spreadsheet of .xls file.
HSSFCell	High level Representation of a cell in a row of a spreadsheet of .xls file.
XSSFWorkbook	Similar to HSSFWorkbook.
XSSFSheet	"
XSSFRow	"
XSSFCell	"

Parameterize tests using Excel sheet

// Path from Where the Exel file has to be read.

String filePath = System.getproperty("user.dir") + "\\ Cred.

// file input stream which needs the input as the file location.

~~FileInput Stream~~

FileInputStream fileStream = new FileInputStream (filepath);

// Workbook Reference of the Exel file.

XSSFWorkbook Workbook = new XSSFWorkbook (fileStream);

// Sheet with needs to be accessed from within the workbook

XSSFSheet Sheet = Workbook.getSheet ("Sheet1");

// Count the Number of Rows.

int rowCount = Sheet.getLastRowNum() - Sheet.getfirstRowNum()

// Get the Current Row where data has to be written.

Row new_Row = getsheet.getRow(i);

⇒ Sheet1.getRow(1).getcell(0).getNumericCell value();

⇒ Sheet1.getRow(0).get(cell(0).getstring

⇒ Sheet1.getRow(1).(reatecell(0).setcellvalue(Str 2);

to Set the values in xls sheet.

```
for(int i=1; i<= rowCount; i++){
    //Get the current row where the data has to be written
  Row newRow = getSheet.getRow(i);
    // Create a new Cell withe reference to the row
  Cell cell = newRow.createCell(2);
    // Set the value in the cell
  Cell.SetcellValue(Dashbord Message);
}
```

To close

⇒ // create an output stream with the ~~reference~~ location where the file has to be ~~credited~~ Created.

```
FileOutputStream fileOutputStream = new FileOutputStream(filePath)
                                                          path
  // write workbook.
workbook.write(fileOutputStream);
  // Close the workbook.
  workbook.close();
```

Take Screenshot in Selenium

①

```
TakesScreenshot ts = ((TakesScreenshot)driver);
File scrfile = ts.getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(scrFile, new File("locationpath\\Image.jpg"));
```

---

②

```
File img = ((TakeScreenshot)driver).getScreenshotAs(OutputType.FILE)
FileUtils.copyFile(img, new File("Path\\ image.png");
```

Junit Test Suite

Junit Test Suite allow us to aggregate all the test Cases from multiple classes to one place and run them together.

TestSuite is used to run the all Class by the SuiteClasses anotation.

Exa Package -/

TakScreenshot.java
login.java,
output.java
windohandling.java
testSuite.java

testSuite

```
@ RunWith (Suite.class)
@ Suite Classes ({ TakeScreenshot.class, login.class, output.class})
  public class test suite {
  }
```

→ after the click on Run, it will Run all the classes.

TestNG Framework

Annotations

- @ BeforeMethod
- @ AfterMethod
- @ BeforeClass
- @ AfterClass
- @ BeforeTest
- @ AfterTest
- @ BeforeSuite
- @ AfterSuite

Execution priority

- → Before Suite method
- → Before Test method
- → BeforClass method
- → Before Method
- → Test 1 method
- → After method
- → Before Method
- → Test2 method
- → After method
- → After Class method
- → After ~~Test~~ method
- ~~→ After Test method~~
- → After Suite method.

- Firstly chose TestNg class
- write the TestNg anotations
- Run that by adding testNg Library
- Rune by the xml file.

Data provider

Ist

```
@DataProvider (name = "credentials")
Public Object[][] getData() {
    Object[][] data = new Object[3][2];
```

No. of Sets of data. — No. of Parameter which you want to pass

```
    data [0] [0] = "donhere";
    data [0] [1] = "don@123";
    data [1] [0] = "donhere1";
    data [1] [1] = "don@123";
    data [2] [0] = "donhere2"
    data [2] [1] = "don@123"
    return data;
}
```

```
@Test (dataProvider = "credentials")
public void f (String username, String password) {
    System.out.println (username);
    System.out.println (password);
}
```

for both Ist & IInd

IInd

```
@DataProvider (name = "credentials")
Public Object[][] data()
{
    return new Object[][]
    {
        {"username1", "password"},
        {"username2", "password"},
        {"username3", "password"}
    };
}
```

Parameters in TestNg

To open the browser by giving the input th the xml file.

```
@ BeforeClass
@Parameters ({"browser", "url"})
  void setup (String browser, String app)
  {
    if (browser.equalsIgnoreCase ("chrome"))
    {
     System.setProperty ("webdriver.chrome.driver", "C:// Path")
     driver = new ChromeDriver();
    }
   else if (browser.equalsIgnoreCase ("firefox"))
    {
    System.setProverty ("Webdriver.gecko.driver", "C://path")
    driver := new FirefoxDriver();
    }

  driver.get(app);   // URL
  }

// Now Create the xml file.

  <Suite name = "All Test Suite">
    <parameter name = "browser" value = "chrome" />
                                        firefox
    <Parameter name = "url" value = "https://login.com"/>
    <test name = "Parameterstest">
        <classess>
            <class name = "Parameters Example"/>
        </classess>
    </test>

  </Suite>
```

(1) (29)

@ Test (priority = 1 , enabled = false) { } //not executed

@ Test (pribrity = 0 , ~~enabled = pass~~) { } /execute at first.

(2) Groups @ Test (groups = { "car" , "ecar" }) ;