**<---Selenium--->**

**Contents:**

1. **Introduction**
2. **Selenium IDE**
3. **Locators**
4. **Web driver Introduction**
5. **Dropdown**
6. **Drag & Drop**
7. **Mouse hover**
8. **Radio & Checkbox**
9. **File upload**
10. **TestNG**
11. **Screenshot**
12. **POM**
13. **Alert**
14. **Frames**
15. **Window handling**
16. **Data driven Framework**
17. **Object Repository**
18. **Jenkins**
19. **Maven**
20. **Reports**
21. **Calendar/Web table**
22. **Wait**
23. **Exception Handling**
24. **java**
25. **Selenium grid**
26. **Collections**
27. **Introduction:**

**Used to automate Web based applications.**

**It is an open source tools.**

**It supports various o/s environments (QTP supports only Windows o/s)**

**Windows**

**Linux**

**Macintosh**

**It supports various browsers**

**Firefox(Default)**

**Chrome (Need Chrome Driver)**

**Internet Explorer (Need IE Driver)**

**It supports various languages (QTP supports only VB script)**

**Java**

**C Sharp**

**Python**

**Perl**

**Ruby**

**PHP**

1. **Selenium IDE**

**It is a Firefox add ins.**

**Supports only Firefox.**

**Not used now a days.**

**Selenium IDE commands are called as Selenese.**

**Only using Element locators and Selenese commands, No Programming.**

**We can Export these commands from File menu to various programming language.**

**55 and Above Firefox version there is no selenium Ide add ins, since they have stopped.**

**Installation:**

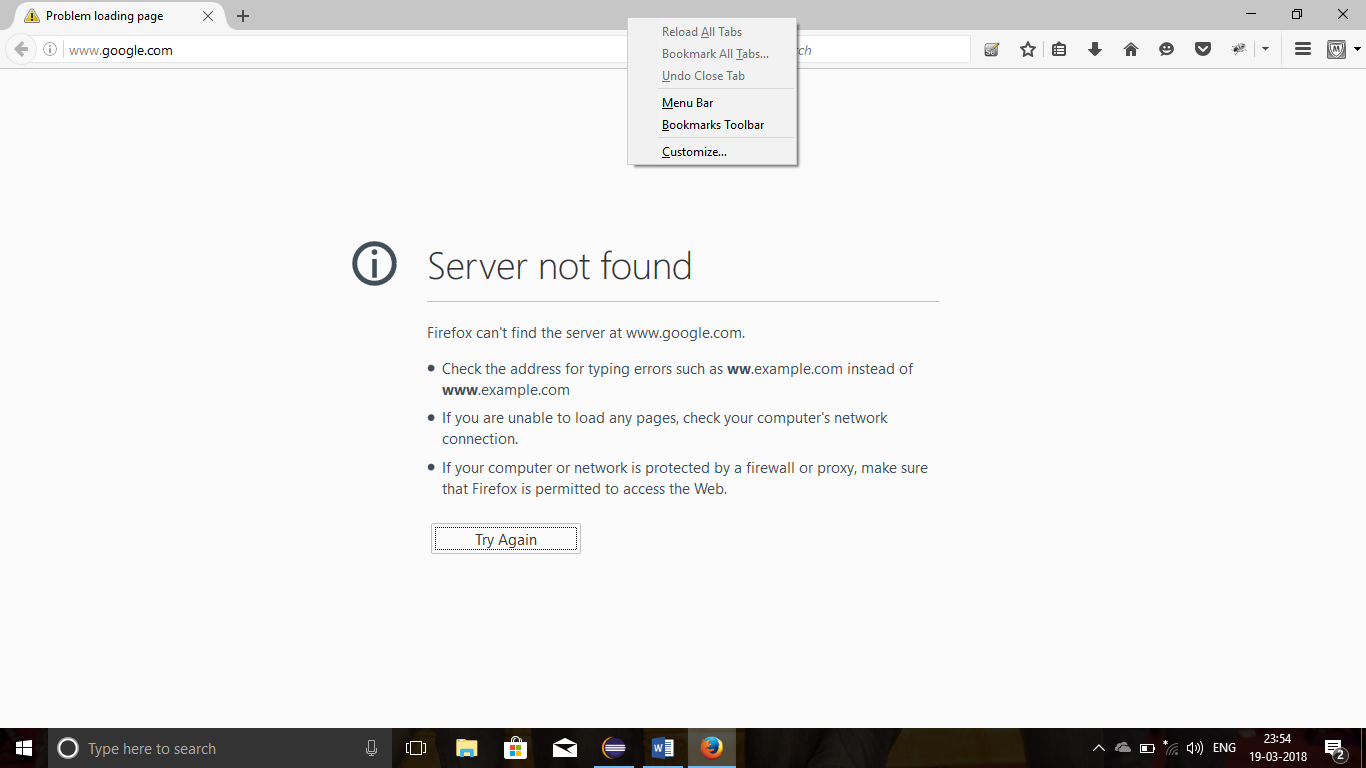
**Download Firefox v 47.0.1 (Which is standard for Selenium)**

**In Firefox browser type selenium IDE and search,**

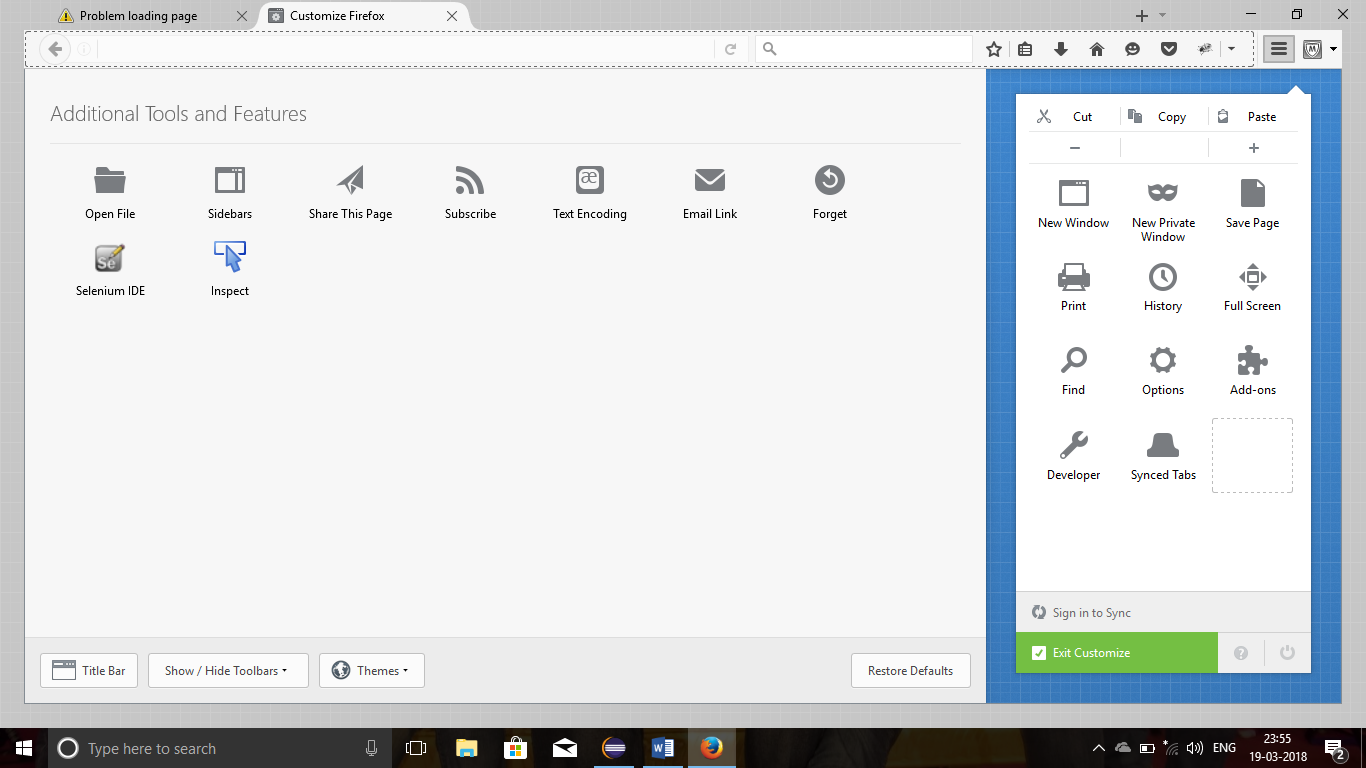
**It will display selenium IDE add ins, click download and install.**

**To check:**

**Right click on the top pane and click Customize,**



**We can see the downloaded Selenium Ide in the Additional tools and features window,**



1. **Environmental setup: WebDriver**

**Download JDK 8 (which contains JRE) in oracle website, install it,**

**Set Enviroinment Path for java, go to my computer->right click->Properties->advanced systemsetting->advanced tab->enviroinmental variable button.**

**New-> “path” in variable name and copy the java bin path in variable location, to verify the installation type JAVA in run command prompt.**

**Download Eclipse and save it in a folder, no need to install just launch it.**

**Download Selenium standalone Server 2.53.1 jar file and add it in eclipse project -> configure build path -> java build path->add external jar file.**

**Firefox version 47.0.1 download and install**

**TestNG , go to help ->market place in eclipse, type TestNG and install it, eclipse will restart, verify it by windows-> preferences in eclipse Oxygen OR for other versions of eclipse go to help-> Install new software’s-> click add-> type TestNG and in below field type** [**WWW.beust.com/Eclipse**](http://WWW.beust.com/Eclipse)**, TestNG will appear and install it.**

**http://newtours.demoaut.com/**

1. **Locators:**

**General Locators:**

**Simple syntax for ID and Name.**

**Syntax:**

**driver.findElement(By.name("value"));**

**driver.findElement(By.id ("value"));**

**driver.findELement(By.linkText(“value”));**

**Example:**

**driver.findElement(By.name("btnLogin"));**

**driver.findElement(By.id("email\_1"));**

**driver. findELement(By.linkText(“Contact us”));**

**X-path:**

**two types:**

**1. Relative X-path**

**2. Obsolete X - Path**

**single attribute:**

**Syntax:**

**//tagname[@Attribute='Value']**

**//tagname[text()= ‘value’] ->for link text**

**Example:**

**//input[@id='text']**

**//div[@type='password']**

**//input[@class='password']**

**//input[@name='password']**

**//input[text()=’Contact Us’] -> for link text**

**Multiple Attribute:**

**General Syntax:**

**//tagname[@attribute=’value’][@attribute=’value’] and so on,**

**Example:**

**//input[@name='username'][@size='20']**

**//input[@type='password'][@name='password'][@id=’email’]**

**Or:**

**General Syntax:**

**//tagname[@attribute=’value’ or @attribute=’value’] - -> Should match any one criteria**

**Example:**

**//input[@name='userName' or @size='20'] - Should match any one criteria name or size**

**And:**

**General Syntax:**

**//tagname[@attribute=’value’ and @attribute=’value’] - -> Should match both criteria**

**Example:**

**//input[@name='userName' and @size='20'] - Should match both criteria name and size**

**Contains:**

**General Syntax:**

**//tagname[contains(@Attribute,'Value')]**

**Ex:**

**//input[contains(@id,'hone\_')]**

**Startswith:**

**General Syntax:**

**//tagname[starts-with(@id,'Value')]**

**Example:**

**//input[starts-with(@name,'ph')]**

**x-path - axes:**

**Types:**

**Following**

**Preceding**

**Following:**

**General syntax:**

**//tagname[@attribute='value']//following::unknown element tagname[index]**

**\*predefined element should be unique by using single, multiple, and, or**

**\*unknown element tagname= Required element tag name**

**Example:**

**//input[@id='name\_3\_firstname']//following::input[1]**

**\*Index 1 = next element of predefined element**

**\*Index 2 = second element of predefined element**

**preceding:**

**General Syntax:**

**//tagname[@attribute='value']//preceding::unknownelementtagname[index]**

**Example:**

**//input[@id='name\_3\_firstname']//preceding::input[1]**

**CSS – Cascading Styles Sheet:**

**We can use the CSS selector in internet explorer since no X-path search engine is not available in the internet explorer.**

**No multiple attributes:**

**Types:**

**id**

**class**

**single attribute**

**contains**

**starts with**

**Ends with**

**ID:**

**General Syntax:**

**#idvalue**

**Example:**

**#email\_1**

**Driver.findElement(By.CSS Selector(#email)**

**Class:**

**General Syntax:**

**.Class Value**

**Example:**

**.username**

**\*use . in the class while using CSS selector.**

**Example:**

**If Class value is Class = email password int**

**We should use like**

**.email.password.int**

**Single Attribute:**

**General Syntax:**

**Tagname[attribute=’Value’] ->no @ used here**

**Example:**

**Input[ID=’email\_1’]**

**Contains:**

**General Syntax:**

**Tagname[attribute\*=’Value’] ->no @ used here**

**Example:**

**Input[class\*=’email\_1’]**

**Starts With:**

**General Syntax:**

**Tagname[attribute^=’Value’] ->no @ used here**

**Example:**

**Input[class^=’email\_1’]**

**Ends With:**

**General Syntax:**

**Tagname[attribute$=’Value’] ->no @ used here**

**Example:**

**Input[class$=’email\_1’]**

1. **Driver Setup:**

**Two Categories:**

FireFox Driver.

Other Drivers(like chrome, internet explorer…)

**FireFox Driver:**

For Firefox we don’t need to download any drivers, because selenium default browser is firefox.

We just type the firefox syntax.

**Example:**

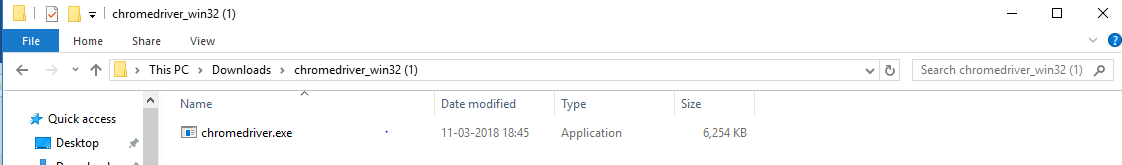
WebDriver driver = **new** FirefoxDriver();

**Other Driver:**

**For example Chrome is used here.**

**Four steps:**

**First download chrome driver(either 32 or 64 bit) and save it in the local folder.**



**Second Set the system property path for chrome driver.**

System.*setProperty*("webdriver.chrome.driver","C:\\Users\\LakshmananDeepthy\\Downloads\\chromedriver\_win32 (1)[\\chromedriver.exe](file:///\\chromedriver.exe)");

**Third use the chrome driver as like Firefox driver**.

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

**Fourth , for Chrome driver we need to maximize the window in genral.**

driver.manage().window().maximize(); // Window maximize syntax , Use Minimize() to minimize the window.

Note:

There is another Driver for Firefox called Gecko Driver

**It is used to Upgraded firefox version**

**We can use the Selenium Standalone server 3.0 versions**

**How to use:**

**Download the Gecko Driver and set path as like Chrome Driver, we can use it**

1. **Drop Down:**

**By using SELECT class.**

**Three steps:**

**First we need to create a WebElement to find the required dropdown.**

WebElement country = driver.findElement(By.*id*("dropdown\_7"));

**Second use the SELECT class**

Select country\_s = **new** Select(country); // create new Select class variable use WebElement name

**Third select the drop down value by Index or by Value or by Visible Text.**

country\_s.selectByIndex(4) // use the Select class variable

**Example:**

**By Index:**

WebElement country = driver.findElement(By.*id*("dropdown\_7"));

Select country\_s = **new** Select(country);

country\_s.selectByIndex(4); //while using index no double quotes

**By Value:**

WebElement month = driver.findElement(By.*id*("mm\_date\_8"));

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

month\_s.selectByValue("3");

**By visible text:**

WebElement date = driver.findElement(By.*id*("dd\_date\_8"));

Select date\_s = **new** Select(date);

date\_s.selectByVisibleText("5");

1. **Mouse Hover:**

**By using Actions class.**

**Three steps:**

First we need to create a WebElement for the required field(Where we need to perform Mouse hover action)

WebElement pim = driver.findElement(By.*id*("menu\_pim\_viewPimModule"));

Second use ACTIONS class

Actions mh = **new** Actions(driver); // Create new Actions class variable and use driver variable not WebElement name.

**Third Use Mousehover general syntax.**

Type Mouse Hover operation syntax.

mh.moveToElement(pim).build().perform(); // Syntax to perform Mouse Hover action.

**Example:**

// Mouse hover using Actions Class.

WebElement pim = driver.findElement(By.*id*("menu\_pim\_viewPimModule"));

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

mh.moveToElement(pim).build().perform();

1. **Radio and CheckBox**

**Radio Button:**

**Three Steps**

**First use List:**

we need to store all the elements in a list.

List<WebElement> radios =driver.findElements(By.*xpath*("//input[@type='radio']"));

**Second use the Advanced for loop:**

Create new WebElement in a for loop and store the values in it from List WebElement.

**for**(WebElement rad:radios)

{

String Val = rad.getAttribute("value"); //We need to get the attribute values for each element and store it a string variable, the values are string so we used String here

**Third Use If condition:**

To check the stored attribute values are equal to our manually entered value

**if**(Val.equalsIgnoreCase("Married"))

{

rad.click();

**break**; // we shoud use break here to stop the loop.

}

}

**Example:**

List<WebElement> radios =driver.findElements(By.*xpath*("//input[@type='radio']"));

**for**(WebElement rad:radios)

{

String Val = rad.getAttribute("value");

**if**(Val.equalsIgnoreCase("Married"))

{

rad.click();

**break**;

}

}

**Checkbox:**

**Same as radio Button, Here we select multiple check boxes.**

**Three Steps**

**First use List:**

we need to store all the elements in a list.

List<WebElement> Che =driver.findElements(By.*xpath*("//input[@type='checkbox']"));

**Second use the Advanced for loop:**

Create new WebElement in a for loop and store the values in it from List WebElement.

**for**(WebElement cb:che)

{

String Val = cb.getAttribute("value"); //We need to get the attribute values for each element and store it a string variable, the values are string so we used String here

**Third Use If condition:**

To check the stored attribute values are equal to our manually entered value

**if**(Val.equalsIgnoreCase("Married"))

{

cb.click();

**break**; // we shoud use break here to stop the loop.

}

}

**Example:**

List<WebElement> checkboxes =driver.findElements(By.*xpath*("//input[@type='checkbox']"));

**for**(WebElement cb:checkboxes)

{

String Val = cb.getAttribute("value"); //

**if**(Val.equalsIgnoreCase("cricket ")) /\* For Multiple selections - If val.equalsIgnoreCase("reading")) || Val.Equals.ignorecase("Cricket") and no break.\*/

{

cb.click();

**break**;

}

}

}

**For Multiple checkbox selection:**

**Only one step:**

**In the if condition we can give multiple values;**

**Note:**

**No break is used here:**

**Example:**

**if**(Val.equalsIgnoreCase("cricket ") || Val.equalsIgnoreCase("Tennis") || Val.equalsIgnoreCase("FOOTBALL "))

{

cb.click();

}

1. **Screenshot:**

**By using TakeScreenshot**

**Screenshot is a File , So we need to use FILE here**

**Three steps:**

First write the syntax for Screenshot and give the output type = file:

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

Second:

Use the FileUtils.CopyFile method and enter the location that where our screenshot file need to save.

FileUtils.*copyFile*(ss, **new** File("F:\\screenshot\\error.png"));

THIRD:

Need to ask prem:

System.***out***.println( FileUtils.*sizeOf*(**new** File("F:\\screenshot\\error.png")));

**Example:**

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

FileUtils.*copyFile*(ss, **new** File("F:\\screenshot\\error.png"));

System.***out***.println( FileUtils.*sizeOf*(**new** File("F:\\screenshot\\error.png")));

1. **Wait:**

**Three categories.**

1. Thread.sleep
2. Implicit Wait
3. Explicit Wait
4. Thread.sleep

It is not advisable, since it delays our code.

Syntax:

Thread.Sleep(5000);

1. Implicit Wait.

We can use default to all programs at the beginning after the getting URL.

Example:

driver.get("http://automationpractice.com/index.php");

driver.manage().timeouts().implicitlyWait(15,TimeUnit.***SECONDS***);

1. Explicit Wait:

**By using webDriverWait**

It is used to tell the code to wait for the particular element .

We can use it any where.

Three Steps:

1. First we need to enter the main syntax next to driver setup.

Example:

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

//explicit Wait:

WebDriverWait wait = **new** WebDriverWait(driver,60);

1. Second We need to create a WebElement for the required element.

Example;

WebElement fn = driver.findElement(By.*id*("menu-item-374"));

1. Third we need to mention in the code that where we need to explicitly wait by using create WebElement with following syntax.

Example:

wait.until(ExpectedConditions.*visibilityOf*(fn));

So that the code waits until the visibility of required element.

Overall Example:

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

//explicit Wait:

WebDriverWait wait = **new** WebDriverWait(driver,60);

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

WebElement fn = driver.findElement(By.*id*("menu-item-374"));

//use wait class where ever we need.

wait.until(ExpectedConditions.*visibilityOf*(fn));

driver.findElement(By.*id*("menu-item-374")).click();