UPES

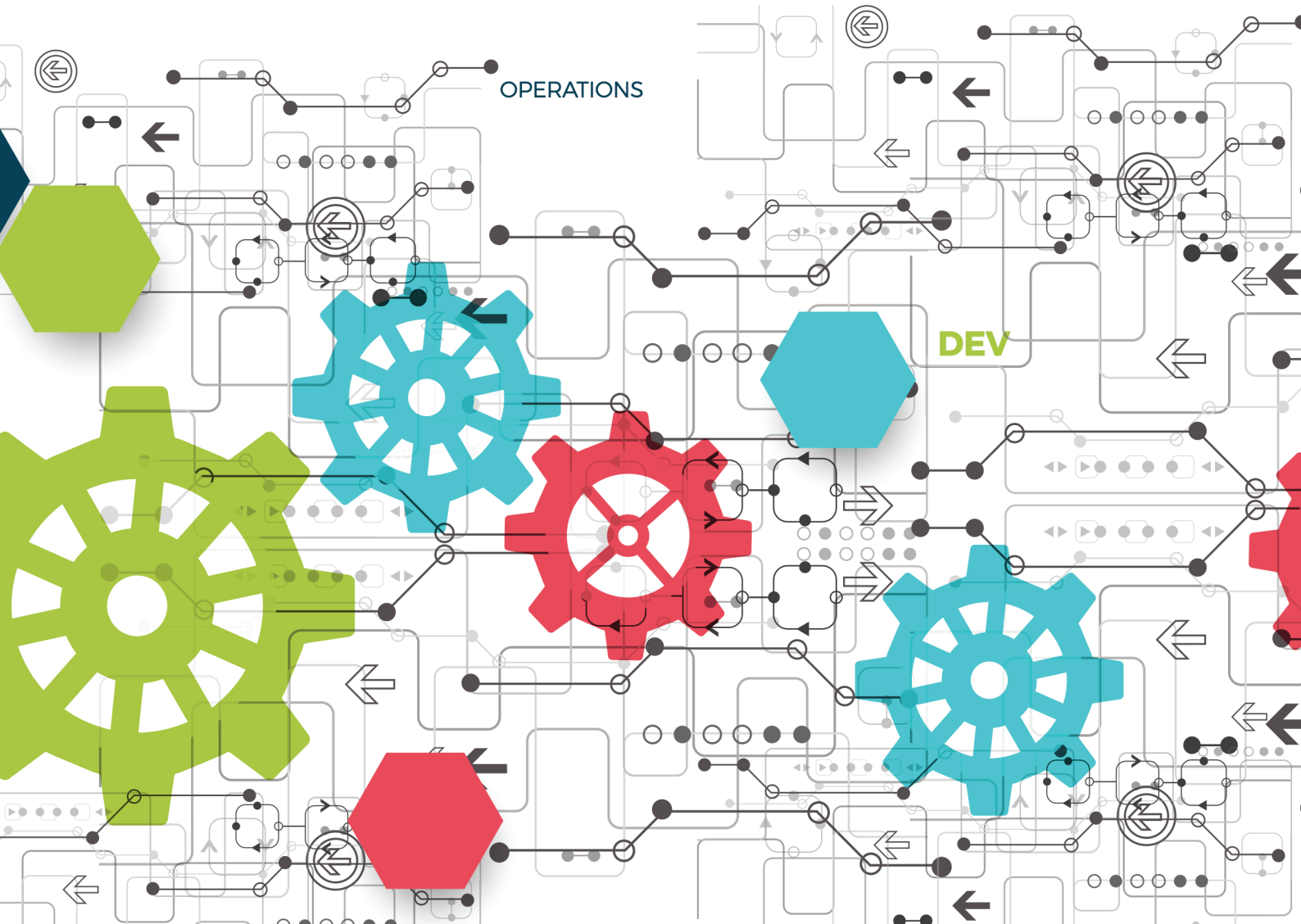
Xebia

B.Tech Computer Science and Engineering in DevOps

Test Automation

MODULE 3

Introduction to Selenium 3.x



Contents

MODULE 3

Introduction to Selenium 3.x

Module Objectives

At the end of this module, you will be able to learn:

- Describe Selenium 3.x advantages and implementation.
- Define drivers for Firefox, IE, chrome, Iphone, Android etc.
- Analyse first Selenium Code.
- Differentiate between Close and Quit.
- Describe Firepath and firebug Add-ons installation in Mozilla.
- Inspect elements in Mozilla, Chrome and IE.
- Identifying WebElements using id, name, class.
- Generate own CssSelectors.
- Differentiate between performance of CssSelectors as compared to Xpaths.
- Define class attribute.
- Handle Dynamic objects/ids on the page.
- Analyse whether object is present on page or not,

You will be informed about the module objectives.

At the end of this module, you will be able to learn:

- Describe Selenium 3.x advantages and implementation.
- Define drivers for Firefox, IE, chrome, Iphone, Android etc.
- Analyse first Selenium Code.
- Differentiate between Close and Quit.
- Describe Firepath and firebug Add-ons installation in Mozilla.
- Inspect elements in Mozilla, Chrome and IE.
- Identifying WebElements using id, name, class.
- Generate own CssSelectors.

- Differentiate between performance of CssSelectors as compared to Xpaths.
- Define class attribute.
- Handle Dynamic objects/ids on the page.
- Analyse whether object is present on page or not,

Module Topics

Let us take a quick look at the topics that we will cover in this module:

- Selenium 3.x advantages and implementation.
- Downloading WebDriver Jars and configuring in eclipse.
- Drivers for Firefox, IE, chrome, Iphone, Android etc.
- First Selenium Code.
- Working with chrome and IE.
- Close and Quit –Difference.
- Firepath and firebug Add-ons installation in Mozilla.
- Inspecting elements in Mozilla, Chrome and IE.
- Various locator strategies.
- Identifying WebElements using id, name, class.
- Finding Xpaths to identify.
- Absolute and Relative Xpaths.
- Creating customized Xpaths without firebug.
- CSS Selectors.
- Generating own CssSelectors.
- Performance of CssSelectors as compared to Xpaths.
- Objects with same id/xpath/cssSelector.
- What is class attribute?
- Handling Dynamic objects/ids on the page.
- Working with different browsers without changing code.
- Managing Input fields, Buttons and creating custom xpaths.
- Managing/Identifying Links with xpaths/css selectors.
- Extracting More than one object from a page.
- Extracting all links of a page/Bulk extraction of objects.
- Finding whether object is present on page or not.

The most major difference between Selenium WebDriver 2 and 3 is that **all the major browser vendors will ship their own WebDriver implementations** (Apple, Google, Microsoft, and Mozilla) instead of being developed by Selenium project (Selenium 2.x) and because they know their browsers better than anyone else, their WebDriver implementations can be tightly coupled to the browser, leading to a better testing experience for all of us.

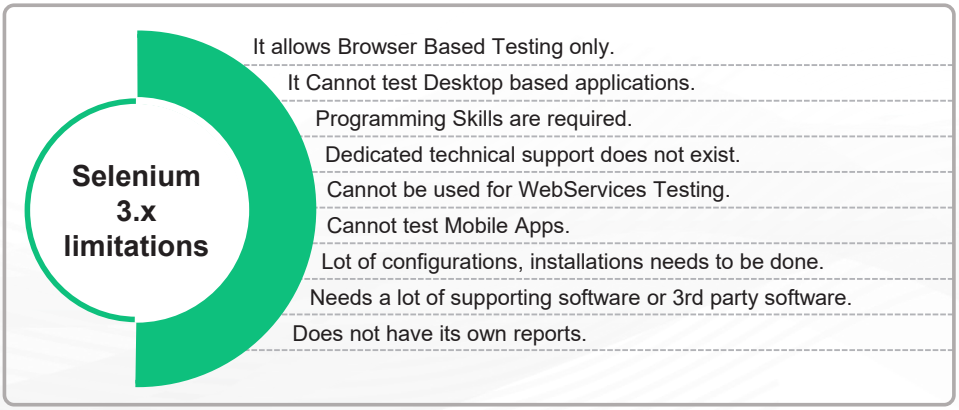
In other words, Selenium 3 will operate Web Elements with native calls (as Selenium 2) executed by the driver, but, this time, browser drivers are developed by the browser vendors themselves.

You will learn about the following topics in this module:

- Selenium 3.x advantages and implementation.
- Downloading WebDriver Jars and configuring in eclipse.
- Drivers for Firefox, IE, chrome, Iphone, Android etc.
- First Selenium Code.
- Working with chrome and IE.
- Close and Quit –Difference.
- Firepath and firebug Add-ons installation in Mozilla.
- Inspecting elements in Mozilla, Chrome and IE.
- Various locator strategies.
- Identifying WebElements using id, name, class.
- Finding Xpaths to identify.
- Absolute and Relative Xpaths.
- Creating customized Xpaths without firebug.
- Generating own CssSelectors.
- Performance of CssSelectors as compared to Xpaths.
- Objects with same id/xpath/cssSelector.
- What is class attribute?
- Handling Dynamic objects/ids on the page.
- Working with different browsers without changing code.
- Managing Input fields, Buttons and creating custom xpaths.
- Managing/Identifying Links with xpaths/css selectors.
- Extracting More than one object from a page.
- Extracting all links of a page/Bulk extraction of objects.
- Finding whether object is present on page or not.

1.3.1 Selenium 3.x limitations

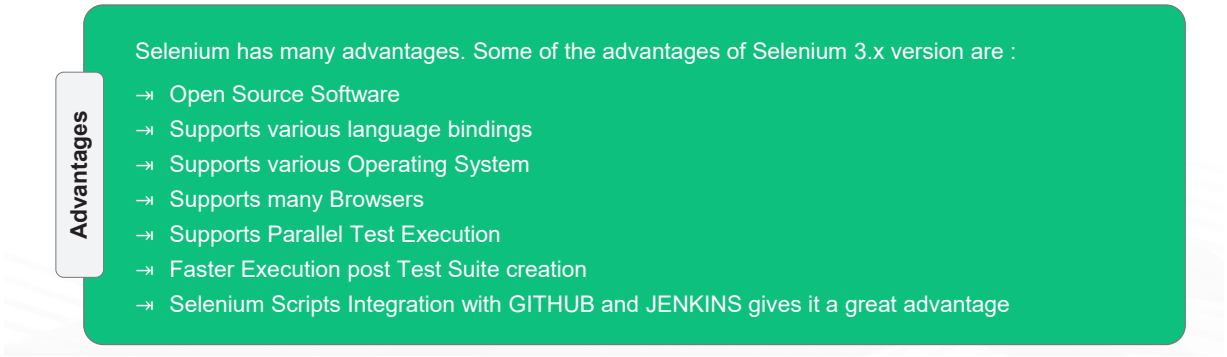
Before we discuss about the advantages, first let's learn about the limitations



Listed below are some disadvantages of Selenium:

- Selenium supports testing of Browser Based Application (i.e., Web-Based testing only).
- Selenium does not support Desktop based Applications (for e.g., we cannot test Calculator, MS Paint, Clock, etc..).
- Selenium requires high skill set in Programming Language. A normal Manual Tester who wishes to upgrade to Automation Testing using Selenium will not find it easy.
- Since Selenium is an open source tool, so there is no technical support or grievance redressal body of some sort. We can only depend on Community Forums and Online Blogs and opinions can be quite diverse in nature.
- Cannot do WebServices Test like SOAP or REST.
- It is dependent on plug-ins like JUNIT and TestNG for reporting and better representation of Validations as internal reporting within Selenium does not exist and internal Validation Criteria for representation is not impressive and cannot be projected to the end client.
- Setting up of Environment Variables, Importing Dependencies, Installing Dependent Softwares and other installations is a cumbersome process which needs prior knowledge and perfection else a small human error will lead to no result and re-work needs to be done.
- Selenium Cannot Automate Captcha image or BarCode.
- Selenium' results are highly dependent on the Application's performance behavior which can vary because of multiple factors (like Page Load Duration, Internet Speed, Complication of DOM Structure, Browser Compatibility) – so the results can be ambiguous and different from one another in every iteration of test.
- Mobile Applications cannot be tested using Selenium.

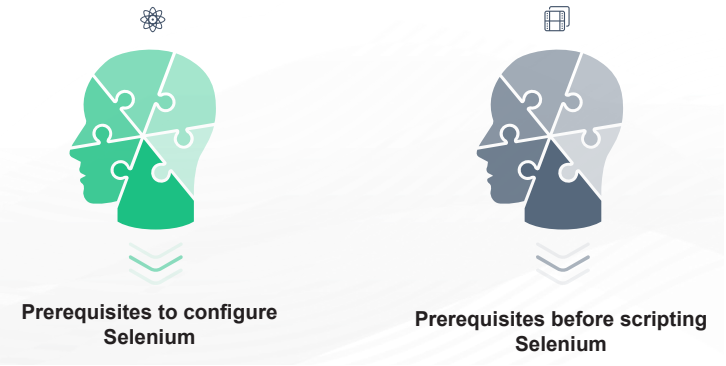
1.3.1 Selenium 3.x advantages



- Selenium is an open source software so it is free like other Automation Tools such as QTP or Test Complete, etc. Even the updated versions are free. Users can configure the same once the release is stable and do not have to pay for anything.
- Selenium supports various language bindings like Java, C#, Perl, Python, PHP, Ruby, JavaScript. Unlike QTP which supports only Visual Basic, Selenium has lots of support for all these languages. Though it is the most stable with JAVA and 90% of the Selenium Testers use JAVA
- The best part of Selenium is we can create a test suite in one environment and run the same in another. Let's say for example, you create the Scripts in Windows and you want to run the same in Mac OS then it can be executed with certain changes.
- Selenium supports Browsers like Firefox, Chrome, IE, Edge, Safari, Opera, etc but it is most stable with Firefox and Chrome.
- Selenium with the powerful feature of Parallel Execution can run multiple test cases in multiple browsers at the same time and help in time saving.
- Initially, setting up Selenium and writing up the Scripts might take time, lot of logic and understanding but that remains the same in case of Manual Testing as well.The distinct advantage of Selenium is that once everything is setup, subsequent executions just happens way faster and results and reports are generated which are quite good looking and also way better in terms of presentation and that is quite important in a professional setup with high profile clients.
- Selenium scripts can easily be pushed and pulled with certain credential validations into various Object Repositories and CI/CD tools too. That helps quite a lot to make everything work in a remote environment. This helps to access the code from anywhere and does not really depend on the physical device.

1.3.1 Selenium 3.x implementation

The Selenium 3.x implementation requires the following:



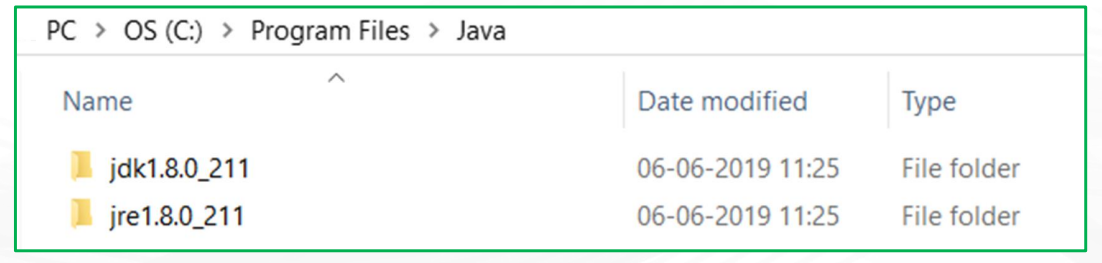
Since we are learning Selenium with JAVA, so we shall learn how to download, install and configure java in our local machine.Then, we will learn from where and how to download the Selenium JAR files which will aid us in using Selenium.We will learn how to download executable files of respective drivers and we shall understand how to use them.

Then we will learn how to import all these JAR files into an Eclipse Project.Finally, we shall use the executable files of the browsers and write a script to understand how it works

1.3.1 Selenium 3.x implementation – Download and Install Java

Prerequisites required to work with Selenium:

Download and install Java:



The steps to download and install JAVA (Windows 10 and Windows 7):

- Open Google and type ‘download jdk 8’
- Please click on the first link as it appears
- It shall redirect you to oracle.com website and it will be under the ‘Downloads’ tab
- Scroll down a bit and you can see ‘Java SE Development Kit 8u231’
- Accept the License Agreement – check this radio button
- As per your operating system and processor bit download the file
- It will ask you for username and password. In case you already have a username and password then use it
- Enter Username and password.

- Once done it will start the download.
- Once download is complete click on the downloaded file and install it
- Once it is installed check in Program Files you can see a Java folder and it shall have a jdk and jre folder respectively.

1.3.1 Selenium 3.x implementation – Configure Java

- Setup Environment Variables for Java
- Verify that Java has been successfully installed.

```
C:\Users\nepl>java -version
java version "1.8.0_231"
Java(TM) SE Runtime Environment (build 1.8.0_231-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.231-b11, mixed mode)
```

```
C:\Users\nepl>javac
Usage: javac <options> <source files>
where possible options include:
  -g                         Generate all debugging info
  -g:none                    Generate no debugging info
  -g:{lines,vars,source}     Generate only some debugging info
  -nowarn                    Generate no warnings
  -verbose                   Output messages about what the compiler is doing
  -deprecation               Output source locations where deprecated APIs are used
  -classpath <path>          Specify where to find user class files and annotation processors
  -cp <path>                 Specify where to find user class files and annotation processors
  -sourcepath <path>         Specify where to find input source files
  -bootclasspath <path>      Override location of bootstrap class files
  -extdirs <dirs>            Override location of installed extensions
  -endorseddirs <dirs>       Override location of endorsed standards path
  -proc:{none,only}          Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery proce
ss
  -processorpath <path>      Specify where to find annotation processors
  -parameters                Generate metadata for reflection on method parameters
  -d <directory>             Specify where to place generated class files
  -s <directory>             Specify where to place generated source files
  -h <directory>             Specify where to place generated native header files
  -implicit:{none,class}     Specify whether or not to generate class files for implicitly referenced files
  -encoding <encoding>       Specify character encoding used by source files
  -source <release>          Provide source compatibility with specified release
```

- Right click on My Computer / This PC > Properties > AdvancedSystemSettings.
- Click on Advanced System settings and then another window System Properties shall appear.
- Click on ‘Advanced’ tab in this window.
- There will be a button ‘Environment Variables’. Click it.
- ‘Environment Variables’ window will open.
- It shall have 2 sections – ‘User variables’ and ‘System Variables’.
- Under the section ‘User variables’ click on the ‘New… ‘ button .
- ‘New User Variable’ window opens with 2 editable sections – ‘Variable name’ and ‘Variable vaule’.
- Enter JAVA_HOME in ‘Variable name’ .
- Enter the path of the jdk in ‘Variable vaule’. In my case the path is C:\Program Files\Java\jdk1.8.0_231.
- Then go to the other section ‘System Variables’ and choose the ‘Path’ variable.
- Please be very careful with the ‘Path’ variable – if you tamper anything un-necessary than your OS might get corrupted.
- Follow my instructions strictly. First I will tell about Windows 7 users and then will tell about Windows 10 users.

- So for windows 7 users – once you select the 'Path' variable, then click on the Edit... Button below..
- You will see a new window opens whose name is 'Edit System Variable' and there will be two editable sections 'Variable name' and 'Variable Value'.
- At the end of the 'Variable Value' – take your cursor and check if there is a semi-colon present (;).
- If semi-colon(;) is not present, then give a semi-colon(;) and paste the path of the jdk bin file. In my caste the path is C:\Program Files\Java\jdk1.8.0_231\bin.
- At the end give a semi-colon(;) and then click on OK and come out.Do not do unnecessary things here.
- For Windows 10 users – select the 'Path' system variable and click on 'Edit...' button and a different 'Edit Environment Variable' button will come with lot of Editable sections and lots of buttons on the right of this window.
- Select New and just paste the jdk bin – C:\Program Files\Java\jdk1.8.0_231\bin and click on OK.
- Then press OK and acknowledge all OKs and that's it.

To make sure JAVA has been installed, follow the following steps:

- To make sure java has been successfully installed, please open a command prompt [Windows+R, cmd]
- Type the command java –version
- Type the command javac
- So you can make sure by the above 2 commands that java has been installed successfully in your system

Note : In case you have not done the above steps correctly then you will not be able to get the correct response in command prompt rather you will get some error messages

1.3.2 Downloading WebDriver jars

Downloading the JAR Files

- Download jars from https://selenium.dev/downloads/
- Download the JARS from 'Selenium Client & WebDriver Language Bindings' against the specific language binding

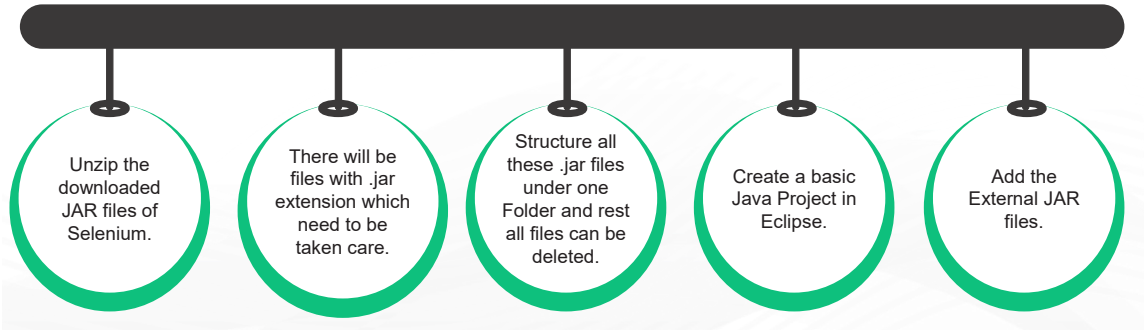
Almost till December 2019, the Selenium website had a different address and it was 'seleniumhq.org' but very recently it got transitioned to 'selenium.dev'. For us the most important tab in this website is the Downloads tab and we will use it more frequently than not to get all our required setup.

Remember Selenium is not downloaded.Selenium is configured. Here are the steps to required to configure Selenium

- Open the URL- https://selenium.dev.

- Click on the 'Downloads' tab.
- Scroll down till you see the section 'Selenium Client & WebDriver Language Bindings'.
- You will come across various language bindings like Ruby, JavaScript, Java, C#, Python.
- We are learning Selenium with Java. So to the right of the Java language you can see under LINKS column there will be a Download link. Click on it .
- Your Selenium JAR files will get downloaded in zipped format.

1.3.2 Downloading WebDriver jars and configuring in Eclipse



The zipped file which has all the Selenium JARS can be unzipped and it will create multiple folders and it will have numerous files which are not of any use to Selenium Testers. We are mostly concerned with .jar files which are present in that.

Here we are listing down same. Create a separate folder and name it as 'SELENIUM' and in that SELENIUM folder make another sub-folder 'JAR Files'.

These .jar files can be unziped. Cut each one of them and add it in the 'JAR Files' folder.

- Now open Eclipse and make a simple Java Project.
- Right Click on the Project and Click on Properties.
- Select 'Java Build Path' > 'Libraries' > Add External JARs…
- This will take to the folder directory of windows. Just go to the JAR files folder and select all the 'jar files and add them.
- Click on Apply and Close.
- Then go and expand the Java Project and refresh it once. You will see a folder known as 'Referenced Libraries'. If you expand it you can see all the jar files in this section.

1.3.3 Drivers for Firefox, IE, chrome, Edge, etc

- Go to https://selenium.dev/downloads/
- Scroll down to the section 'Platforms Supported by Selenium'.

- You will see Browsers and Operating Systems.
- Expand Browsers.
- Click on the documentation hyperlink mentioned for Firefox, Chrome, Safari, etc.
- Let's say you open for Chrome browser. You can see there will be all versions available in Downloads.
- Download the exe file as per the Browser version already in your local machine.
- Save that exe file in a folder as it will be a good structural approach.
- Similarly you can download for other browsers and save the executable file in the same folder as mentioned in point no 8.

1.3.4 First Selenium Code

Steps to create the first Selenium code:

- Create a java project in Eclipse.
- Import all the selenium jar files in the project.
- Make a package inside the project and then make a class.

FirstSeleniumCode.java

```
1  package selenium;
2
3  import org.openqa.selenium.WebDriver;
5
6  public class FirstSeleniumCode {
7
8      public static void main(String[] args) {
9
10         System.setProperty("webdriver.chrome.driver", "C:\\SELENIUM\\BROWSERDRIVER\\chromedriver.exe");
11         WebDriver driver = new ChromeDriver();
12         driver.get("https://www.google.com");
13
14     }
15
16 }
```

- Create a Java Project, here we have created SeleniumLearning
- Create a package inside it, here we have named it as selenium
- Create a class inside it, here we have named it as FirstSeleniumCode
- In the main method of this class, write a basic code.

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

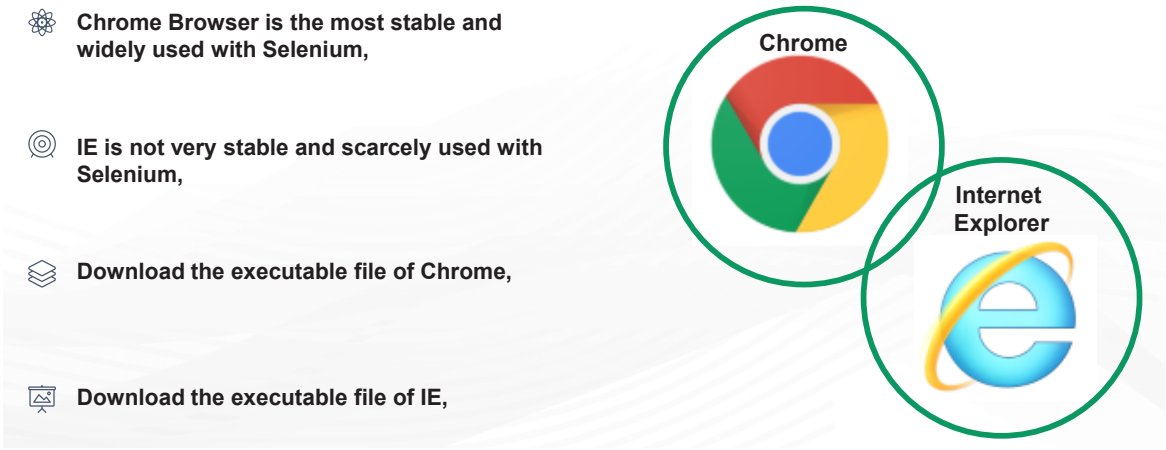
WebDriver driver = new ChromeDriver();

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

- The above code will invoke the Chrome Browser and it will open Google website in the Chrome Browser window.
- System – is a class which contains several methods and it cannot be instantiated.
- setProperty – is a method which has a String key and String value .
- "webdriver.chrome.driver" – this is the String key.

- "C:\\SELENIUM\\BROWSERDRIVER\\chromedriver.exe" – this is the String value and in this case it is the path of the chrome driver executable which is kept in my local machine.
- WebDriver is an interface.
- driver is an object reference.
- new is the keyword used to create objects in java.
- ChromeDriver() – is a class, it controls a Chrome browser running on the local machine.
- new ChromeDriver() – this is the object.
- driver.get(https://www.google.com") – this allows the object reference driver to access get method to open the URL mentioned.

1.3.4 Working with Chrome and IE



ChromeDriver is standalone server that implements the W3C WebDriver standard. ChromeDriver is available for Chrome on Android and Chrome on Desktop (Mac, Linux, Windows and ChromeOS). The ChromeDriverServer executable must be downloaded and placed in your PATH.

The following system properties (read using System.getProperty() and set using System. setProperty("webdriver.chrome.driver")

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

WebDriver driver = new ChromeDriver();

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

The InternetExplorerDriver is a standalone server which implements WebDriver's wire protocol. The IEDriverServer executable must be downloaded and placed in your PATH

The following system properties (read using System.getProperty() and set using System. setProperty("webdriver.ie.driver").

System.*setProperty("webdriver.ie.driver", "C:\\SELENIUM\\BROWSERDRIVER\\iedriver. exe");*

WebDriver driver = new InternetExplorerDriver();

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

What Did You Learn So Far ?

1. Why is Selenium more popular than QTP or for that matter any other automation testing tool ?
   - **A) Selenium is open source and free**
   - **B) Supports many language bindings**
   - **C) Supports multiple browsers**
   - **D) none of the above**
   - **E) a, b and c are true**

Answer: 1 : (e)

2. Which is the Browser driver in Selenium ?
   - **A) FirefoxDriver**
   - **B) ChromeDriver**
   - **C) htmlUnitDriver**

Answer: 2 : (c)

3. What is the browser executable driver for Firefox ?
   - **A) chromedriver.exe**
   - **B) firefoxdriver.exe**
   - **C) geckodriver.exe**

Answer: 3 : (c)

1.3.4 Close and Quit - Difference

*FirstSeleniumCode.java

```
1  package selenium;
2  import org.openqa.selenium.WebDriver;
3  import org.openqa.selenium.chrome.ChromeDriver;
4
5  public class FirstSeleniumCode {
6
7  public static void main(String[] args) {
8
9          System.setProperty("webdriver.chrome.driver", "C:\\SELENIUM\\BROWSERDRIVER\\chromedriver.exe");
10         WebDriver driver = new ChromeDriver();
11         driver.get("https://www.google.com");
12         driver.close();
13
14     }
15
16 }
```

FirstSeleniumCode.java

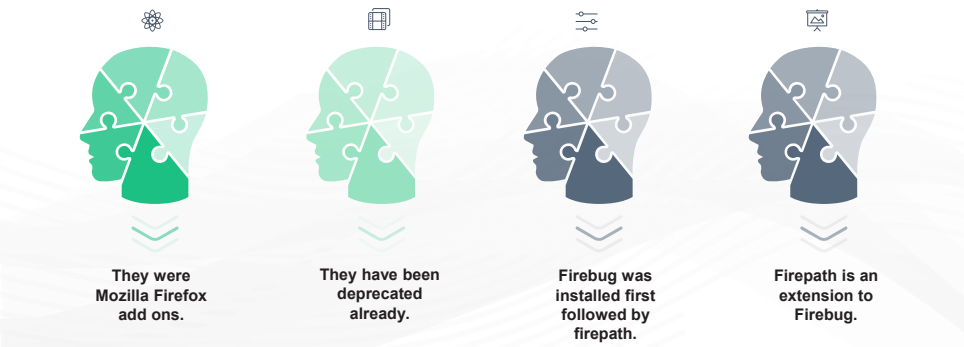
```
1  package selenium;
2  import org.openqa.selenium.WebDriver;
3  import org.openqa.selenium.chrome.ChromeDriver;
4
5  public class FirstSeleniumCode {
6
7  public static void main(String[] args) {
8
9          System.setProperty("webdriver.chrome.driver", "C:\\SELENIUM\\BROWSERDRIVER\\chromedriver.exe");
0          WebDriver driver = new ChromeDriver();
1          driver.get("https://www.google.com");
2          driver.quit();
3
4      }
5
6  }
7
```

quit() – this method is used to end the running WebDriver instance. By calling this method, it closes all the active browser windows(which were initiated by the current driver instance) and the active session ends**.**

close() - method is used to close the browser which is currently in focus.

1.3.4 Firebug and Firepath add-ons installation in Mozilla

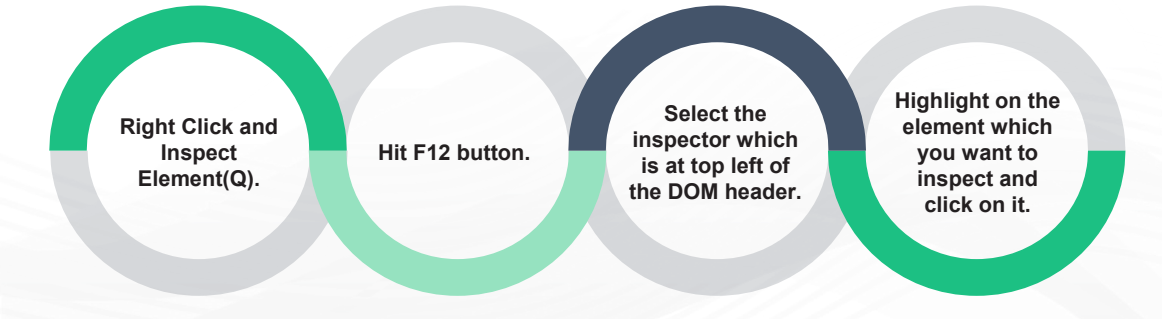
Key points about Firebug and Firepath add-ons:



Both Firebug and Firepath were used to highlight elements in the webpage. And have to install old version of Firefox and install it.

1.3.5 Inspecting Elements in Mozilla

Ways to inspect Elements in Mozilla :-



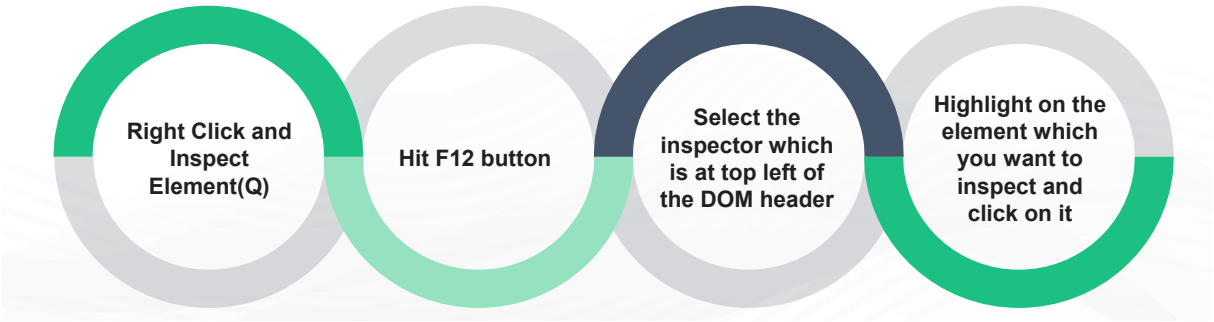
To open the DOM Structure just hit F12 button.

To highlight a particular element in the DOM there are 2 ways:-

- Right Click on that element and select ‘Inspect Element (Q) and it will get highlighted in the DOM.
- Click on F12 and select the inspector which is at the top left of the DOM header and then highlight the element in the webpage and click on it.

1.3.5 Inspecting Elements in Chrome

Ways to inspect Elements in Chrome:-



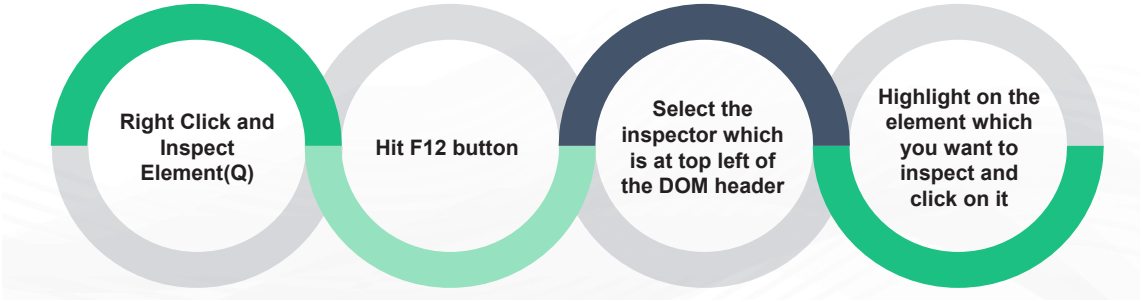
To open the DOM Structure just hit F12 button

To highlight a particular element in the DOM there are 2 ways:-

- Right Click on that element and select ‘Inspect Element (Q) and it will get highlighted in the DOM
- Click on F12 and select the inspector which is at the top left of the DOM header and then highlight the element in the webpage and click on it

1.3.5 Inspecting Elements in IE

Ways to inspect Elements in IE :-



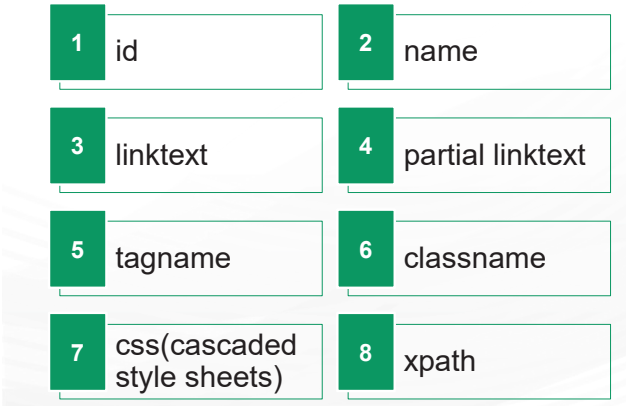
To open the DOM Structure just hit F12 button.

To highlight a particular element in the DOM there are 2 ways:-

- Right Click on that element and select 'Inspect Element (Q) and it will get highlighted in the DOM.
- Click on F12 and select the inspector which is at the top left of the DOM header and then highlight the element in the webpage and click on it.

1.3.6 Various Locator Strategies

Selenium gives the user 8 options to locate elements



In real time projects, there are certain rules and those rules are followed in terms of element location. The highest priority is given to id. In case, id is absent then next priority will be given to classname and then next priority will be given to name.

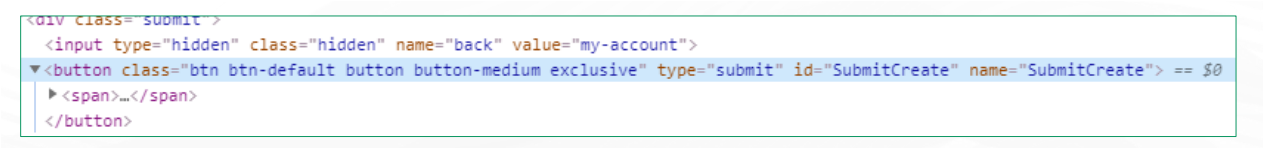
Sometimes, all 3 of them can be used and sometimes only 2 can be used.Linktext, partial linktext – are scarcely used as links can change and so can their text.

Css selectors and xpath – knowledge on them and using them is really fun. It gives a Selenium Tester lot of confidence if someone can master these two locator strategies.

1.3.6 Identifying Web Elements using id

Steps for identifying Web Element using id:

- Id is given the highest priority and preference.
- Id is genuinely safe and generally unique.
- Id is the fastest locator option.
- Dynamic ids are also possible – in this case Selenium Testers have to be careful



In the above screenshot, there is a html tag button, the value of id attribute is “SubmitCreate” and there are also name and class attributes.

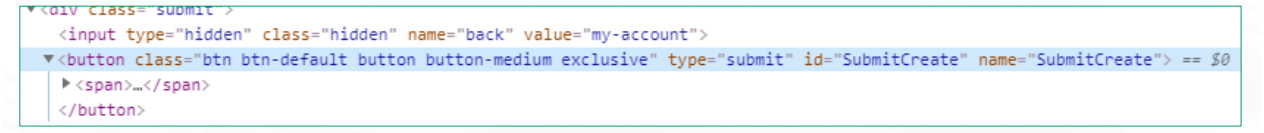
Observe closely, the value of class is multiple. Hence, it is highly risky to use the class; even though name is just a single value and it can be used, but still prefer to use id -in case of DOM while traversing to the element. If there are multiple elements with the same configuration then adjust accordingly.

Selenium script will be: driver.findElement(By.id(“SubmitCreate”));

1.3.6 Identifying Web Elements using name

Steps for identifying Web Element using name:

- This option should be used in case id is not available.
- Names are quite repetitive in a DOM structure.
- Hence, only unique name should be considered.



In the above screenshot, for html tag button, the value of name attribute is “SubmitCreate” and there are also id and class attributes.

There are chances that name is not unique. To determine the uniqueness of any attribute, there are certain ways to do in the DOM structure.

Selenium script will be: driver.findElement(By.name(“SubmitCreate”));

What Have We Learnt So Far ?

1. In case F12 does not work and right click is disabled for a website, then how will you inspect any element ?
   - **A) Mark website as secure and cannot be inspected**
   - **B) Use Tools > Developer Tools**
   - **C) Use some other Automation Tool**

Answer: 1 : (b)

2. Under what circumstances, attributes id and class and name are not safe to be used ?
   - **A) id is unique**
   - **B) id ends or starts with a number**
   - **C) multiple class values**
   - **D) b and c are correct**

Answer: 2 : (d)

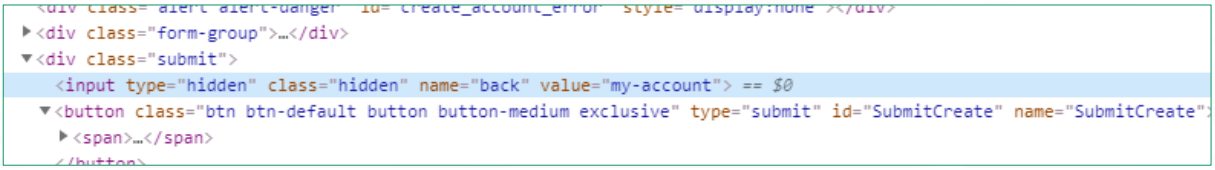
3. Which syntax is correct ?
   - **A) //html[@id = 'somevalue' and @name = 'somevalue']**
   - **B) html[@id = 'somevalue' and @name = 'somevalue']**
   - **C) //html[@id = 'somevalue' and @name = "somevalue"]**
   - **D) //html[@id = 'somevalue' & @name = 'somevalue'']**
   - **E) //html[@id = 'somevalue' && @name='somevalue']**

Answer: 3 : A

1.3.6 Identifying Web Elements using class name

Steps for identifying Web Element using class name:

- This option should be used in case the id and name are not available.
- classname are quite repetitive in a DOM structure.
- Hence, only unique classname should be considered.



In the above screenshot, for the html tag input, the value of classname attribute is “hidden” and there is also name attribute

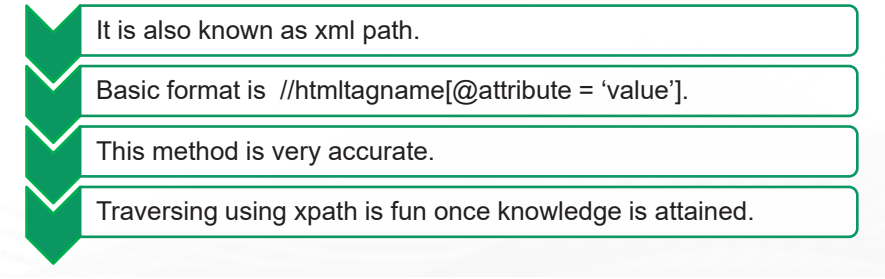
Observe closely the value of class is unique. But actual determination of uniqueness has to be done while we traverse.So unique classname has to be selected.

There are chances that classname is not unique. So determining the uniqueness of any attribute there are certain ways to do in the DOM structure

Here, the Selenium script will be : driver.findElement(By.classname(“hidden”));

1.3.6 Finding Xpaths to identify

Steps for identifying Web Element using Xpaths:



XPath is defined as XML path. It is a syntax or language for finding any element on the web page using XML path expression. XPath is used to find the location of any element on a webpage using HTML DOM structure.

Basic format is //htmltag[@attribute = ‘value’]

Note : Anything which is missing in the syntax will result in nothing. So please follow syntax properly

// - this is selecting current node

htmltag – it is the tag of the particular node

@ - this is for selecting attribute

Attribute – this could be any of the locators which we have discussed like id, name, class, etc

Value – this is the value of the attribute

1.3.6 Absolute Xpath

Key points about Absolute Xpath

- Absolute Xpath is more accurate than Relative Xpath.
- It is direct way to find element.
- Any changes made in the path will result in Xpath failure.Hence,usage of Absolute Xpath in selenium is not advised.
- Syntax starts with single forward slash (/).
- Can select the element from the root node.

Absolute Xpath contains the complete path from the root element to the desirable element. It is highly accurate and does not require too much skill to deduce.

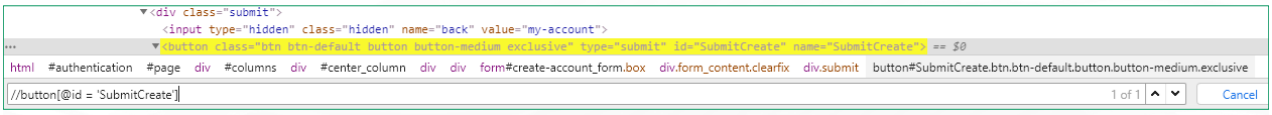
/html/body/div/div[2]/div/div[3]/div/div/div[1]/form/div/div[3]/input[1] – this is an example of Full Xpath or Absolute Xpath.

In the DOM structure, right click on the element which is highlighted and then Copy > Copy Xpath and then you can get the absolute xpath.Strictly not advisable for selenium testers to use it.

1.3.6 Relative Xpath

Key points about Relative Xpath:

- Path starts from the middle of the HTML DOM structure.
- It starts with the double forward slash (//).
- Relative xpath can search the web element anywhere in the DOM.

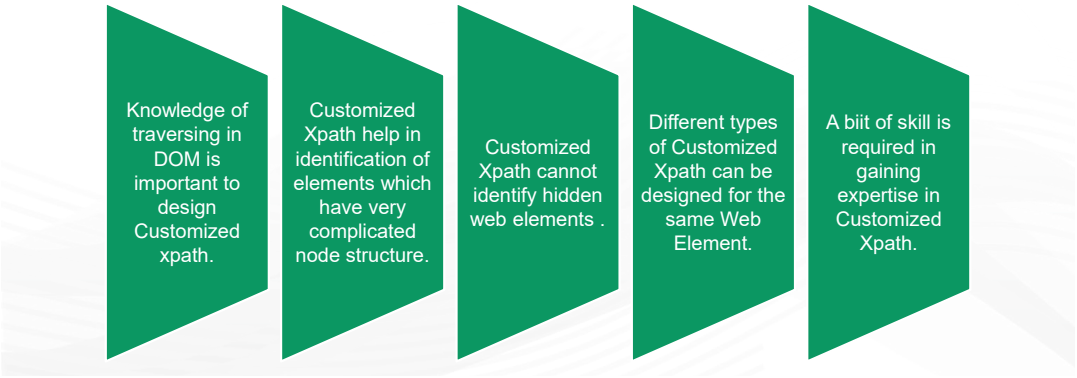


- Xpath syntax - //button[@id = ‘SubmitCreate’]
- Always select the Xpath with result as 1 of 1 as shown in the image in this slide

Relative Xpath will give accurate Xpaths. Even though the accuracy is slightly behind Absolute Xpath but the construction of relative Xpath is such that they are more stable and less prone to change. And if changed, the changes can be handled easily without a lot of rework.

1.3.7 Creating customized xpath without using firebug

Creating customized xpath require the following:



Example of Customized xpaths for any web application:

//htmltag[@attribute = 'value']

//htmltag[@attribute = 'value' and @attribute = 'value']

//htmltag[text() = 'name of the text']

//htmltag[contains(text(), 'name of the text'))]

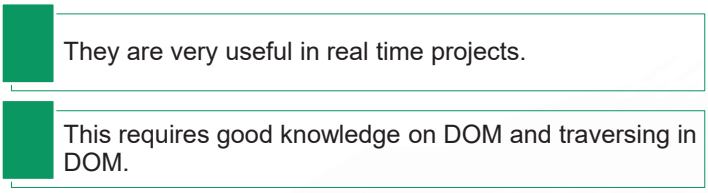
//htmltag[starts-with(@attribute, 'value')]

//htmltag[ends-with(@attribute, 'value')]

Note : You also can use //*[@attribute = 'value'] but avoid using this. Rather specifically mention the html tag.

1.3.7 Creating customized xpath – Dynamic xpath

Dynamic xpath are advanced concepts of Xpath.



This is important as this is where most of the Selenium Testers get stuck because they are not skilled enough to use this. This may give testers a tough time but is very easy and once someone masters this it becomes fun.

The concept of the following:-

- child
- parent
- ancestor
- descendant

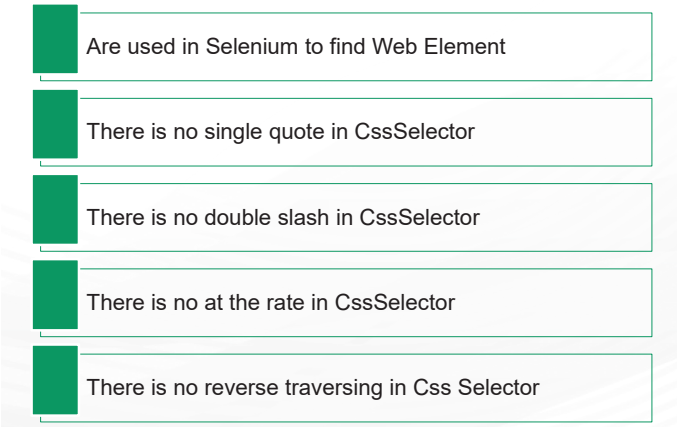
- preceding-sibling
- following-sibling
- preceding
- following
- self

Some examples of Dynamic Xpaths are:-

```
driver.findElement(By.xpath("//div[@id = 'loginContainer']/descendant::div[20]/a[1]")).click();

driver.findElement(By.xpath("//td[contains(text(), 'dummyId@dummySite.com')]/preceding-sibling::td[1]/parent::*/child::td[1]/child::input[@type = 'checkbox' and @name = 'SelectedUsers']"));

driver.findElement(By.xpath("//td[contains(text(), 'prateek@gupta.com')]/preceding::td[1]/parent::*/child::td[1]/child::input[@value = '1034' and @type = 'checkbox' and @name = 'SelectedUsers']"));
```

1.3.8 Css Selectors

Cascading Style Sheet Selectors:



cssSelectors are the fastest Locators to find a Web Element. They are precise too. And they can be customized like Xpaths. There is a change in syntax how they are written

Xpath //html [@attribute = 'value']

Css html[attribute = value]

So from the above you can see the basic syntax difference

Also, in Xpath you can traverse anywhere using preceding, preceding-sibling, ancestor and other ways but that is not applicable to css. You can only move forward.

cssSelectors are widely used in case of List <WebElement> because they act faster and more precisely when it comes to selection of Multiple Elements

1.3.8 Generating Own CSS Selectors

Selector	Example	Example description
.class	.intro	Selects all elements with class="intro"
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element,..	div, p	Selects all elements and all <p> elements

Lots of practice is required to master this. Try these practically.

input[id = UserName]

input[id = UserName][name = UserName]

instead of html[id = value] we can directly use #value

instead of html[class = value] we can directly use .value

.login-form

.login-form>div

.login-form>div:nth-child(1)

.login-form>:nth-child(1)

.login-form>:nth-child(2)

.login-form>:nth-child(2)>input

.login-form>:nth-child(2)>#login1

div[class=login-form] > div:nth-child(2) > input

div[class=login-form]>:nth-child(2)>input

div[class=login-form] > div:nth-child(2) > #login1

shopping

when there are 2 classes then we have to write .classname1.classname2

#homewrapper > div.icondiv > a:nth-child(6) > div > u

div[class=divicon relative]

.divicon.relative

.divcon

.divicon.relative>u

News

#hm_top_navlink_div>div:nth-child(1)

#hm_top_navlink_div>div:nth-child(1)

#hm_top_navlink_div>a:nth-child(3)>div

#hm_top_navlink_div>:nth-child(3)>div

forgot password -

.floatR>a - 2

.f14.margTop10.form-label>div:nth-child(2)>a

.f14.margTop10.form-label>:nth-child(2)>a

.f14.margTop10.form-label>:nth-child(2)>:nth-child(1)

body > div > div.content-area > div.rhs-area.floatR > div.login-area-free.floatR > div > form > div > div.f14.margTop10.form-label > div.floatR > a

Home-

div[class=head-wrapper]>div:nth-child(2)>a- home

.head-wrapper>.floatR>a - home

body > div > div.head-wrapper > div.floatR > a

1.3.8 Performance of css Selectors as compared to Xpaths

Listed below is comparison of CSS Selectors and Xpaths

- Css Selectors perform faster than Xpaths
- Css Selectors are better when the elements are multiple
- Xpath can traverse both backward and forward but Css Selectors can only traverse in forward direction

Preceding and ancestor concepts is not there in CSS Selectors as reverse traversing is not allowed in CSS Selectors.

1.3.9 Objects with same id/xpath/css Selector

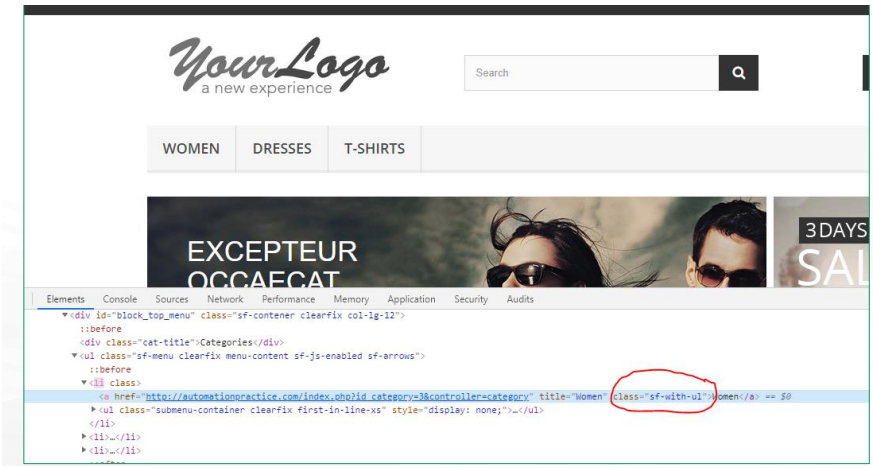
To handle objects which might have same id, use concept of traversing via Xpath.

- ⇥ Open DOM [click on F12 key or right click > Inspect Element]
- ⇥ Ctrl+F in DOM
- ⇥ Write Xpath and make sure it is 1 of 1

The above points mentioned in the slide is how we technically handle objects which might have same id. As Selenium will pick randomly as it encounters hence, by making Xpath more precise, we can solve this issue.

1.3.10 What is Class attribute ?

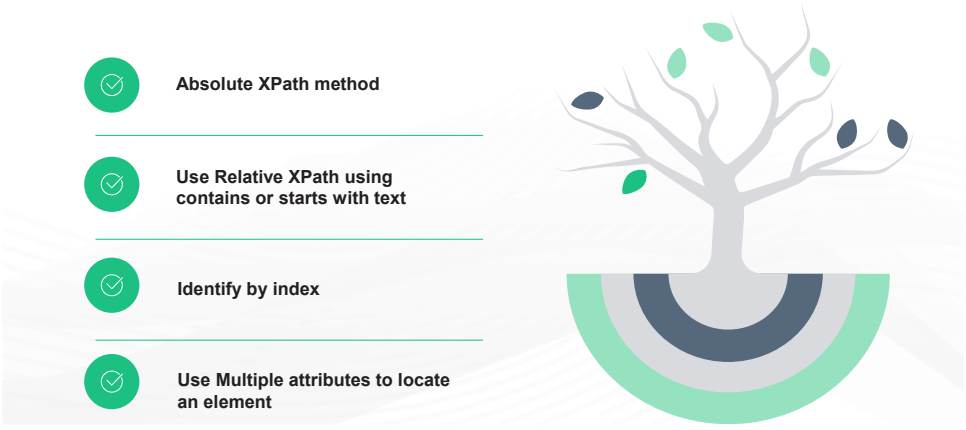
- Class is an attribute inside the DOM which will have a value.
- It is used as a locator identification



Class is an attribute inside the DOM which will have a value. It is used as a locator to identify web elements. In the image, there is class attribute along with its value.

1.3.11 Handling Dynamic Objects/ids on page

To Handle Dynamic Objects/ids on page use the following methods:



- Absolute method might be the easiest but it is the riskiest. Avoid this completely
- Again Relative Xpath is my favorite as well as it is the safest. The only glitch is you should know how to traverse smartly in the DOM using Xpath
- Sometimes, you will have multiple elements with same locator value. For example there may be two submit buttons with id starting with 'Submit'. In this case you can use findElements method and locate the element using the index.
- To identify a particular element you can use multiple attributes if a single attribute is not enough to identify your web element uniquely.

1.3.12 Working with different browsers without changing code

Selenium allows to work with multiple browsers without changing code. Given belwo are steps to attain the same

- Make a Java Project, add Selenium JAR files and add TestNG Library
- Make a package and make a class
- Make a file known as config.properties
- Pass the browser for e.g., browser = chrome and also pass the url = http://automationpractice.com/index.php
- Using TestNG's annotations @BeforeTest, @Test, @AfterTest write the code
- Do not forget to declare a constructor whose body will have the code for the Properties file

Please follow the code below and run this as Run As > TestNg Suite. Create your own package name and package class. If you are copy pasting this code, please use the same package name and class which is provided, Also, the path of the config.properties file in your system will be different. Please be careful.

In case, you want to change from chrome to firefox, you just go and change in config.properties file and save it. You need not change anything in the code.

```
package browsers;


import java.io.FileInputStream;

import java.util.Properties;


import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeDriverService;

import org.openqa.selenium.edge.EdgeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;
```



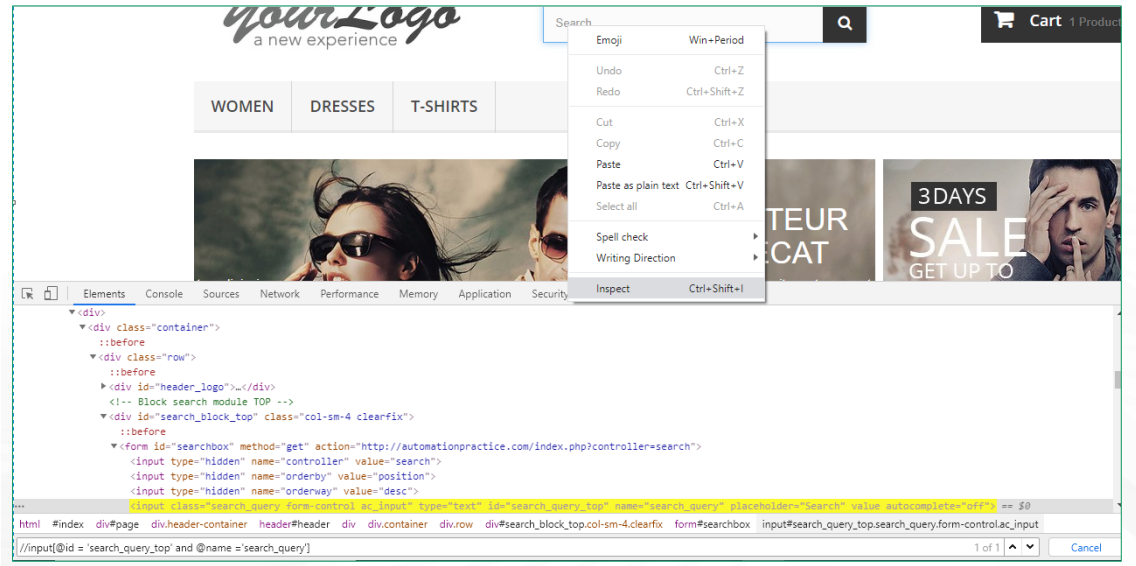
```
import org.testng.annotations.Test;

public class Multiple_Browsers {

public static WebDriver driver;
public static Properties prop;
public static FileInputStream ip;

public Multiple_Browsers() throws Exception {
prop = new Properties();
ip = new FileInputStream("C:/Users/nepl/eclipse-workspace/SELENIUM_WEBDRIVER_CODE/src/test/resources/config.properties");
prop.load(ip);
}

@BeforeTest
public void SetUpBrowser() {
System.out.println("Browser Setup");
}

@Test
public static void InitializeBrowser() {
String browsername = prop.getProperty("browser");

if (browsername.equals("chrome")) {
System.setProperty("webdriver.chrome.driver", "C:\\SELENIUM\\BROWSERDRIVER\\chromedriver.exe");
System.setProperty(ChromeDriverService.CHROME_DRIVER_LOG_PROPERTY, "C:\\SELENIUM\\chromelog.txt");
System.setProperty(ChromeDriverService.CHROME_DRIVER_SILENT_OUTPUT_PROPERTY, "true");
driver = new ChromeDriver();
```

 |

```
driver.get(prop.getProperty("url"));
driver.manage().window().maximize();
}
else if (browsername.equals("firefox")) {
System.setProperty("webdriver.gecko.driver", "C:\\SELENIUM\\BROWSERDRIVER\\geckodriver.exe");
System.setProperty(FirefoxDriver.SystemProperty.DRIVER_USE_MARIONETTE, "true");
System.setProperty(FirefoxDriver.SystemProperty.BROWSER_LOGFILE, "C:\\SELENIUM\\dumphere.txt");
driver = new FirefoxDriver();
driver.get(prop.getProperty("url"));
driver.manage().window().maximize();

}

else if (browsername.equals("MicrosoftEdge")) {
System.setProperty("webdriver.edge.driver", "C:\\Windows\\System32\\MicrosoftWebDriver.exe");
driver = new EdgeDriver();
driver.get(prop.getProperty("url"));
driver.manage().window().maximize();

}
}

@AfterTest
public void tearDown() {
driver.quit();
}

}
```


1.3.13 Managing Input fields, Buttons and creating custom xpaths

Input Field – will have html tag input



Please go to website : http://automationpractice.com/index.php

you can see an input field ‘search’. Right click on it and click on Inspect [you can see that in the above in this slide]

You will get a Blue highlight. I There is a Yellow highlight because we have written the Xpath too. The details are discussed later.

Check the blue highlight and you will see that:-

html tag is ‘input’

class is “search_query form-control ac_input”

id is “search_query_top”

name is “search_query_top”

Now press Ctrl+F

You will see an editable textbar in the bottom of your page

Please write this Xpath - //input[@id = ‘search_query_top’ and @name =’search_query’]

Que : Now how will you enter anything inside the ‘Search’ field ??

Ans : By using method ‘sendKeys’

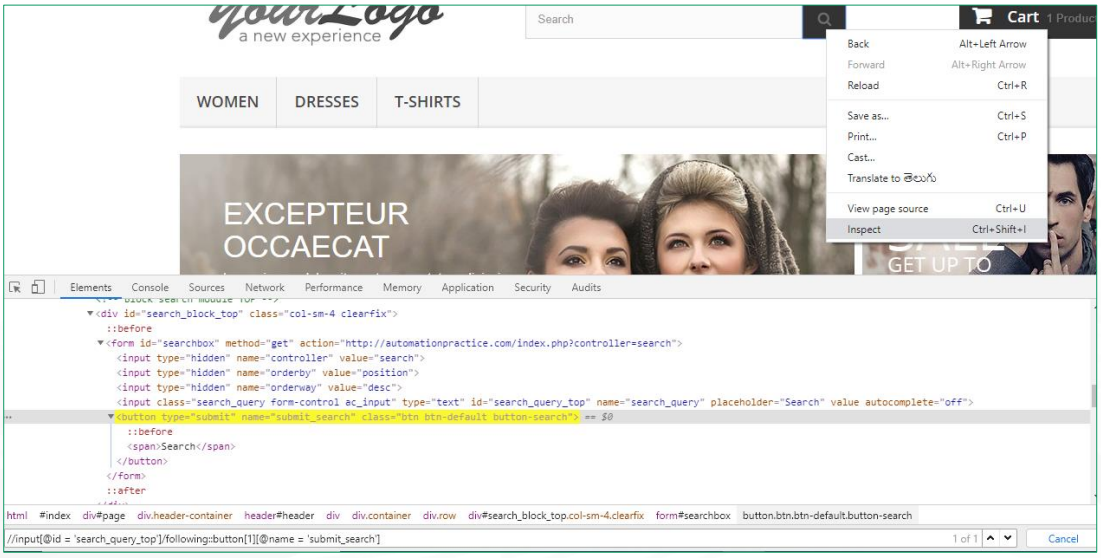
Attach the following code. Please run this code. And as always do not copy paste everything. Make your own package and class and also path of your browser executable.

package normalSeleniumCode;


```
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeDriverService;

public class AutomationPractice {

public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver", "C:\\SELENIUM\\BROWSERDRIVER\\chromedriver.exe");
System.setProperty(ChromeDriverService.CHROME_DRIVER_LOG_PROPERTY, "C:\\SELENIUM\\chrome.log");
System.setProperty(ChromeDriverService.CHROME_DRIVER_SILENT_OUTPUT_PROPERTY, "true");

WebDriver driver = new ChromeDriver();
driver.navigate().to("http://automationpractice.com/index.php");
driver.manage().window().maximize();
driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
driver.findElement(By.xpath("//input[@id = 'search_query_top' and @name ='search_query']")).sendKeys("DRESSES");
}
}
```

Note : Run the above Java Code as a Java Application. Whatever string value you will give inside sendKeys("DRESSES") that will be shown. I have kept "DRESSES". You can give as per your wish.

Buttons – they will have html tag 'button'



Please go to website : http://automationpractice.com/index.php

you can see to the right of input field ‘search’ there is a magnifying glass kind of image under a black color fill. Check for the image above. That is a button

Right click on it and click on inspect.

You can see the highlight in the DOM.

Html tag is ‘button’

We have made a custom xpath - //input[@id = ‘search_query_top’]/following::button[1][@ name = ‘submit_search’]

In case of a button there will always be a method known as click();

Please use the above xpath and also I am writing the code.

driver.findElement(By.*xpath(“//input[@id = ‘search_query_top’]/following::button[1][@name = ‘submit_search’]”)).click();*

```
package normalSeleniumCode;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeDriverService;
public class AutomationPractice {
public static void main(String[] args) {
```


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

System.*setProperty(ChromeDriverService.**CHROME_DRIVER_LOG_PROPERTY, "C:\\SELENIUM\\chrome.log");***

System.*setProperty(ChromeDriverService.**CHROME_DRIVER_SILENT_OUTPUT_PROPERTY, "true");***

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

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

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

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

driver.findElement(By.*xpath("//input[@id = 'search_query_top' and @name ='search_query']")).sendKeys("DRESSES");*

driver.findElement(By.*xpath("//input[@id = 'search_query_top']/following::button[1][@name = 'submit_search']")).click();*

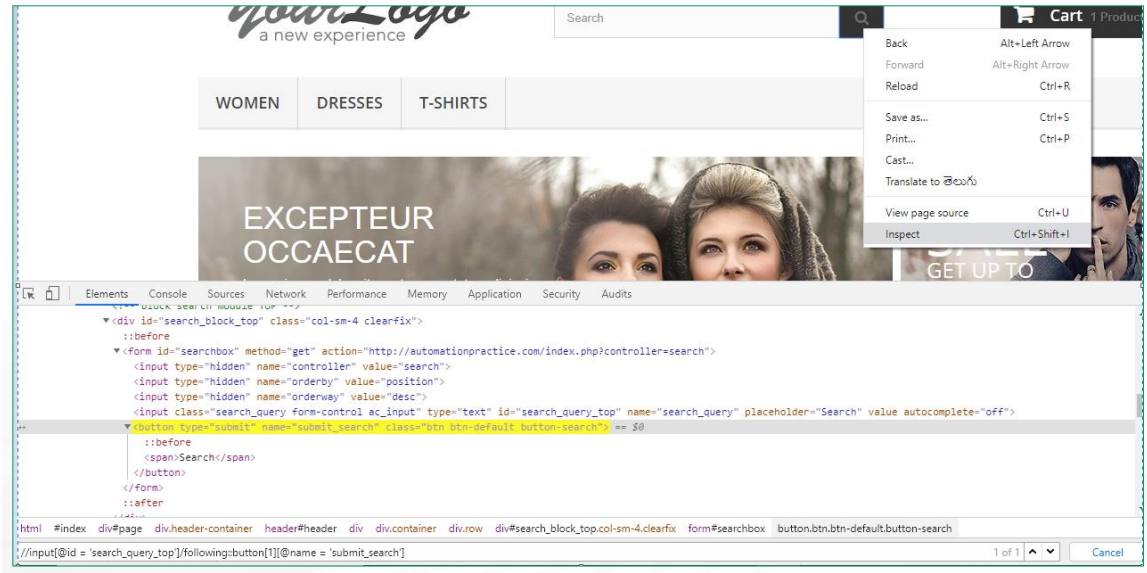
}

}

Note : Run the above Java Code as a Java Application. Whatever string value you will give inside sendKeys("DRESSES") that will be shown. I have kept "DRESSES". You can give as per your wish.. Then the code will allow you to click on the button. Please do this exercise in practical.

1.3.13 Managing/Identifying Links with xpaths/css selectors

Link with cssSelector



Please go to website : http://automationpractice.com/index.php

There will be a link ‘Contact Us’

Right click on it and click on Inspect.

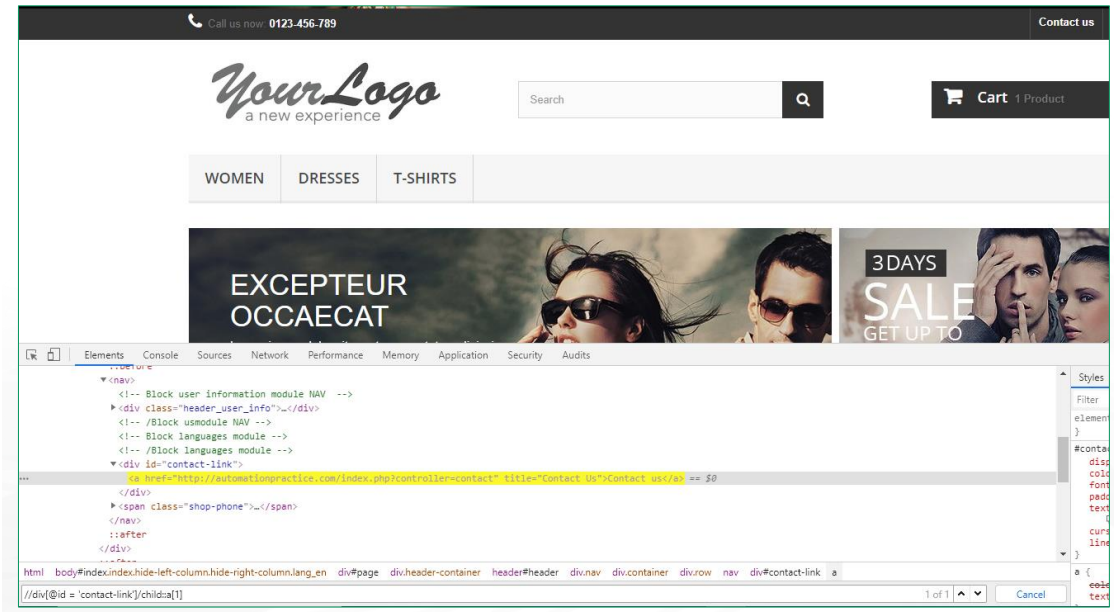
Please write the cssSelector : #contact-link>a:nth-child(1)

The code is shared here:

```
package normalSeleniumCode;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeDriverService;

public class AutomationPractice {

public static void main(String[] args) {

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

System.setProperty(ChromeDriverService.CHROME_DRIVER_LOG_PROPERTY, “C:\\SELENIUM\\chrome.log”);

System.setProperty(ChromeDriverService.CHROME_DRIVER_SILENT_OUTPUT_PROPERTY, “true”);


WebDriver driver = new ChromeDriver();

driver.navigate().to(“http://automationpractice.com/index.php”);

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

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

driver.findElement(By.xpath(“//input[@id = ‘search_query_top’ and @name =’search_query’]”)).sendKeys(“DRESSES”);

driver.findElement(By.xpath(“//input[@id = ‘search_query_top’]/following::button[1][@name = ‘submit_search’]”)).click();

driver.findElement(By.cssSelector(“#contact-link>a:nth-child(1)”)).click();

}

}
```

Note : Run the above Java Code as a Java Application.. Please do this exercise in practical.

Link with Xpath



Please go to website : http://automationpractice.com/index.php

There will be a link ‘Contact Us’

Right click on it and click on Inspect

Please write the xpath : //div[@id = ‘contact-link’]/child::a[1]

I am writing the code below.

```
package normalSeleniumCode;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeDriverService;

public class AutomationPractice {

public static void main(String[] args) {


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

System.setProperty(ChromeDriverService.CHROME_DRIVER_LOG_PROPERTY, “C:\\SELENIUM\\chrome.log”);

System.setProperty(ChromeDriverService.CHROME_DRIVER_SILENT_OUTPUT_PROPERTY, “true”);

WebDriver driver = new ChromeDriver();
```

```
driver.navigate().to("http://automationpractice.com/index.php");
driver.manage().window().maximize();
driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
driver.findElement(By.xpath("//input[@id = 'search_query_top' and @name ='search_query']")).sendKeys("DRESSES");
driver.findElement(By.xpath("//input[@id = 'search_query_top']/following::button[1][@name = 'submit_search']")).click();
driver.findElement(By.xpath("//div[@id = 'contact-link']/child::a[1]")).click();
}
}
```

Note : Run the above Java Code as a Java Application.. Please do this exercise in practical.

1.3.14 Extracting more than one object from a page

- We use findElements which returns a List<WebElement> or an empty list if nothing matches.
- With the reference variable call the size() function and then print this

```
List<WebElement> list = driver.findElements(By.tagName("a"));
System.out.println("total number of links are "+  list.size());
```

Attaching the code below:-

To find the number of links present in the website http://automationpractice.com/index.php?

```
package normalSeleniumCode;
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.openqa.selenium.chrome.ChromeDriverService;
public class AutomationPractice {
public static void main(String[] args) {

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

 |

System.*setProperty(ChromeDriverService.**CHROME_DRIVER_LOG_PROPERTY, "C:\\SELENIUM\\chrome.log");*

System.*setProperty(ChromeDriverService.**CHROME_DRIVER_SILENT_OUTPUT_PROPERTY, "true");*

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

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

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

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

List<WebElement> list = driver.findElements(By.*tagName("a"));*

System.***out.println("total number of links are "+ list.size());***

}

}

}

1.3.14 Extracting all links of a page/Bulk extraction of objects

Extracting total links as well as printing the name of all the links:

```
List<WebElement> list = driver.findElements(By.tagName("a"));
System.out.println("total number of links are "+  list.size());

for(int i=0; i<list.size(); i++) {
System.out.println(list.get(i).getText());
```

Attaching the code below:-

T find the number of links present in the website http://automationpractice.com/index.php? and also printing the name of all the links.

package normalSeleniumCode;

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.openqa.selenium.chrome.ChromeDriverService;


```
public class AutomationPractice {

public static void main(String[] args) {

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

System.setProperty(ChromeDriverService.CHROME_DRIVER_LOG_PROPERTY, "C:\\SELENIUM\\chrome.log");

System.setProperty(ChromeDriverService.CHROME_DRIVER_SILENT_OUTPUT_PROPERTY, "true");

WebDriver driver = new ChromeDriver();

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

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

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

List<WebElement> list = driver.findElements(By.tagName("a"));

System.out.println("total number of links are "+  list.size());

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

System.out.println(list.get(i).getText());

}

}

}
```

To extract all links of a page/Bulk extraction of objects

Use isDisplayed() method

```
List<WebElement> list = driver.findElements(By.tagName("a"));

System.out.println("total number of links are "+  list.size());


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

WebElement link = list.get(i);

System.out.println(link.getText() + "------" + link.isDisplayed());
```

Attaching the code below:-

To find the number of links present in the website http://automationpractice.com/index.php?

Also, printing the name of all the links and finding out which link is present on the page. By executing the below code you can see the output console. The ones with true are present and ones with false are not present.


```
package normalSeleniumCode;
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.openqa.selenium.chrome.ChromeDriverService;
public class AutomationPractice {
public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver",  "C:\\SELENIUM\\BROWSERDRIVER\\chromedriver.exe");
System.setProperty(ChromeDriverService.CHROME_DRIVER_LOG_PROPERTY, "C:\\SELENIUM\\chrome.log");
System.setProperty(ChromeDriverService.CHROME_DRIVER_SILENT_OUTPUT_PROPERTY, "true");

WebDriver driver = new ChromeDriver();
driver.navigate().to("http://automationpractice.com/index.php");
driver.manage().window().maximize();
driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

List<WebElement> list = driver.findElements(By.tagName("a"));
System.out.println("total number of links are "+  list.size());

for(int i=0; i<list.size(); i++) {
WebElement link = list.get(i);
System.out.println(link.getText() + "------" + link.isDisplayed());
}
}
}
```


1.3.15 Finding whether object is present on page or not

Make a function

```
public static boolean isElementPresent(String locator, String locatorType) {
List<WebElement> allElements = null;
```

Call this static function in the main function

```
isElementPresent("//div[@class = 'header_user_info']/child::a[1]", "xpath");
```

Use a while loop

```
while (isElementPresent("//div[@class = 'header_user_info']/child::a[1]",
"xpath")) {
System.out.println("Element is Present");
break;
```

Please find the below code. We have used the 'SignIn' link's xpath in http://automationpractice.com/index.php? to determine whether it is present or not

package normalSeleniumCode;

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.openqa.selenium.chrome.ChromeDriverService;

public class AutomationPractice {

public static WebDriver driver = new ChromeDriver();

public static void main(String[] args) {

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

System.*setProperty(ChromeDriverService.**CHROME_DRIVER_LOG_PROPERTY, "C:\\SELENIUM\\chrome.log");***

System.*setProperty(ChromeDriverService.**CHROME_DRIVER_SILENT_OUTPUT_PROPERTY, "true");***

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

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

 |

```
driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
isElementPresent("//div[@class = 'header_user_info']/child::a[1]", "xpath");
while (isElementPresent("//div[@class = 'header_user_info']/child::a[1]", "xpath"))
{
System.out.println("Element is Present");
break;
}

}

public static boolean isElementPresent(String locator, String locatorType) {
List<WebElement> allElements = null;
if(locatorType.equals("xpath"))
allElements = driver.findElements(By.xpath(locator));
else if(locatorType.equals("cssSelector"))
allElements = driver.findElements(By.cssSelector(locator));
else if(locatorType.equals("id"))
allElements = driver.findElements(By.id(locator));
else if(locatorType.equals("name"))
allElements = driver.findElements(By.name(locator));

if(allElements.size() == 0)
return false;
else
  return true;

}
}
```


In a nutshell, we learnt:

1. Selenium 3.x advantages and implementation
2. Downloading WebDriver Jars and configuring in eclipse
3. Drivers for Firefox, IE, chrome, Iphone, Android etc
4. First Selenium Code
5. Working with chrome and IE
6. Close and Quit –Difference
7. Firepath and firebug Add-ons installation in Mozilla
8. Inspecting elements in Mozilla, Chrome and IE
9. Various locator strategies
10. Identifying WebElements using id, name, class
11. Finding Xpaths to identify
12. Absolute and Relative Xpaths
13. Creating customized Xpaths without firebug
14. Css Selectors
15. Generating own CssSelectors
16. Performance of CssSelectors as compared to Xpaths
17. Objects with same id/xpath/cssSelector
18. What is class attribute?
19. Handling Dynamic objects/ids on the page
20. Working with different browsers without changing code
21. Managing Input fields, Buttons and creating custom xpaths
22. Managing/Identifying Links with xpaths/css selectors
23. Extracting More than one object from a page
24. Extracting all links of a page/Bulk extraction of objects
25. Finding whether object is present on page or not

Now, you have reached the end of the module, In this module, you have learned:

- Selenium 3.x advantages and implementation
- Downloading WebDriver Jars and configuring in eclipse
- Drivers for Firefox, IE, chrome, Iphone, Android etc
- First Selenium Code
- Working with chrome and IE
- Close and Quit –Difference
- Firepath and firebug Add-ons installation in Mozilla
- Inspecting elements in Mozilla, Chrome and IE
- Various locator strategies
- Identifying WebElements using id, name, class
- Finding Xpaths to identify
- Absolute and Relative Xpaths
- Creating customized Xpaths without firebug
- Css Selectors
- Generating own CssSelectors
- Performance of CssSelectors as compared to Xpaths
- Objects with same id/xpath/cssSelector
- What is class attribute?
- Handling Dynamic objects/ids on the page
- Working with different browsers without changing code
- Managing Input fields, Buttons and creating custom xpaths
- Managing/Identifying Links with xpaths/css selectors
- Extracting More than one object from a page
- Extracting all links of a page/Bulk extraction of objects
- Finding whether object is present on page or not

Release Notes

B. TECH CSE with Specialization in DevOps

Semester Six -Year 03

Release Components.
Facilitator Guide, Facilitator Course Presentations, Student Guide, Mock exams and relevant lab guide.

Current Release Version.
1.0.0

Current Release Date.
19 Jan 2020

Course Description.
Xebia, has been recognized as a leader in DevOps by Gartner and Forrester and this course is created by Xebia to equip students with set of practices, methodologies and tools that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

The language used in this course is US English. Our sources of reference for grammar, syntax, and mechanics are from The Chicago Manual of Style, The American Heritage Dictionary, and the Microsoft Manual of Style for Technical Publications.

Bugs reported	Not applicable for version 1.0.0
Next planned release	Version 2.0.0 Jan 2021