Selenium Notes

Wring optimized Codes:

In Java we can reduce number of lines of a code when there is a assignment operator which will be executed only one time.

Ex: 1 Stage 1

```
int i=10;
int j=20;
int k;
k=i+j;
System.out.println(k)
```

O/P: 30

Stage 2

```
int i=10;
int j=20;
System.out.println(i+j);
```

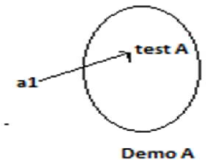
O/P: 30

Stage 3

```
int i=10;
System.out.println(i+20);
System.out.println(10+20);
```

O/P: 30

```
Ex: 2 class DemoA
    {
        public DemoB testA()
        {
        DemoB b1=new DemoB();
        return b1;
        }
    }
    class DemoB
    {
        public void testB()
        {
            System.out.println("Hi");
        }
    }
    //main class

    DemoA a1=new DemoA();
    DemoB b2=a1.testA();
    b2.testB();
        ⇩
    DemoA a1=new DemoA();
    a1.testA().testB(); // optmised code 2
        ⇩
    new DemoA().testA().testB(); //optimised code 1
```




```
Ex 3:
     class System
     {
         static PrintStream out=new PrintStream();
     }
     class PrintStream
     {
         public void println(int i)
         {

         }
         public void println(String s)
         {

         }
     }
     PrintStream p=System.out;
     p.println("qspiders");
              ⇩
     System.out.println("qspiders"); // optmised code
```

Assignment:

1. List any 5 free Automation tool & any 5 Commercial Automation Tool.

Open Source Automation tool:

Name	Tools
FitNesse	Open source
Robot Frawork	Open source
Robotium	Open source
Selenium	Open source
Sikuli	Open source
watir	Open source
Jmeter	Open source
Junit	Open source
Cocumber	Open source
Webcorder	Open source

Commercial Automation tools:

Name	Tools
eggPlant	Testplant
Eiffel Studio Autotest	Eiffel Software
HP Quick Test Professional (QTP)	HP
(RFT) IBM Rational Functional Tester	IBM Rational
Lab view	National Instruments
Maveryx	Maveryx
Rational Robot	IBM Rational
Silk Test	Borland
Soatest	Parasoft
Visual Studio Test Professional	Microsoft
Load Runner	HP

Automation:

Simulation of any work using system or a tool is called "***Automation***".

Example for Automation:

Fan, Mixer Grinder, Microwave, Trimmer, TV, AC, Escalators, Automated parking system, KIOSK Machine, lift etc.

Advantages of Automation:

i. It is faster & saves time.
ii. It reduces the effort
iii. It is reusable
iv. Consistent & Efficient
v. Accurate

Limitations:

i. Initial investment is high.
ii. It requires constant maintenance
iii. Requires additional skill set
iv. 100% automation is not possible (It may be too costly or we may not have the required technology).
Note: Because of above reasons we automate only long term projects, we automate only repeating tasks (Regression testing).

Generally we do not automate

a. Adhoc testing
b. Usability testing
c. Game testing (specially 3D)
d. Verification of Audio, Video clips & Animation.
Automating 60-80% of testcases is considered as good Automation coverage.

Selenium:

It is a free & open source web application automation tool.

Note:

1. **Free**: In order to use selenium also we need not to purchase any license.

2. **Open source**: We can download the source code of selenium software itself so that we can customize it according to our project need.

3. **Web Application**: Using selenium we can test only web application but not client/server & standalone applications.

   Ex: ***Web application***: Facebook, Google etc.
   Client/Server: Skype, Gtalk, Whatsapp
   Standalone: Calculator, Notepad, MS office.

4. **Automation tool**: Selenium is used to test the application features, in this context automation refers to simulation of manual testing steps & tool refers to a software application.

Selenium Versions:

The Selenium tool has following 3 major version.

1. Selenium Core - 2004
2. Selenium RC (Remote Control) – 2006
3. Selenium WebDriver – 2008

Q. What is IDE?

IDE stands for Integrated Development Environment. It is a centralized place which can be used to perform all development related activities such as

1. Creating folder structures
2. Writing the code
3. Compiling the code
4. Debugging the code
5. Executing the code
6. Storing the code in Build repository
7. Versioning the code (Automatically generates the version for the code each time when it is modified).
8. Creating the build
9. Performing Unit testing
10. Deploying the build

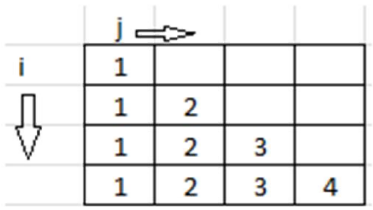
Q. Write a java code to print the following output.

```
1 2 3
1 2 3
1 2 3
```

```
public class DemoA
{
    public static void main(String[] args)
    {
    for(int i=1; i<=3; i++)
    {
        for(int j=1; j<=3; j++)
        {
            System.out.print(j+ " ");
        }
        System.out.println();
    }
    }
}
```

Q. Write a java code to print the following pattern.

```
1
1 2
1 2 3
1 2 3 4
```



```java
public class DemoA
{
    public static void main(String[] args)
    {
    for(int i=1; i<=4; i++)
    {
        for(int j=1; j<=i; j++)
        {
            System.out.print(j+ " ");
        }
        System.out.println();
    }
    }
}
```

Execution:

i	j	SOP(j+" ")	SOPln()
1	1	1_	Enter
2	1	1_	
	2	1_2_	Enter
3	1	1_	
	2	1_2_	Enter
	3	1_2_3_	
4	1	1_	
	2	1_2_	Enter
	3	1_2_3_	
	4	1_2_3_4_	

Q. Write a java code to print the following pattern.

```
1 2 3 4
1 2 3
1 2
1
```

```
public class DemoA
{
    public static void main(String[] args)
    {
    for(int i=4; i>=1; i--)
    {
        for(int j=1; j<=i; j++)
        {
            System.out.print(j+ " ");
        }
        System.out.println();
    }
    }
}
```

Q. Write a java code to print the following pattern.

```
4
4 3
4 3 2
4 3 2 1
```

```
public class DemoA
{
    public static void main(String[] args)
    {
    for(int i=4; i>=1; i--)
    {
        for(int j=4; j>=i; j--)
        {
            System.out.print(j+ " ");
        }
        System.out.println();
    }
    }
}
```

Selenium IDE:

It is a "Record & playback" tool & it is a Add-on for Mozilla Firefox browser.

Selenium IDE internally contains Selenium Core, during runtime it will record steps into java script of selenium core & executes it.

Installing Selenium IDE:

Steps:

i. Open ***Mozilla Firefox*** browser
ii. Go to following website: http://docs.seleniumhq.org/download/
iii. Click on Download latest released version 2.5.0 link which is present under ***Selenium IDE section***.
iv. Click on '***Allow'***.
v. Click on '***Install Now***'
vi. Click on '***Restart Now'***

This will close & reopens the Firefox browser. In the '***Menu bar'*** go to '***Tools'*** & select the option '***Selenium IDE'***.

Note: If the '***Menu bar***' is not visible, then click on ***Firefox Menu*** which is present in left top corner of the browser. Go to ***Web Developer*** & select 'Selenium IDE'.

Installing Selenium IDE offline:

Note: In order to install any Add-on of Mozilla Firefox, first we need to download it & while downloading the Add-ons do not use Firefox browser.

Ex: ***Open IE Browser***, Go to ***Download page of Selenium IDE***, click on '***Selenium IDE***' download link 2.5.0, click Save.

Select the required location such as ***Desktop*** & click on ***Save*** which will download the following file on to the Desktop***: selenium-ide-2.5.0.xpi***

XPI stands for ***Cross Platform Installer***.

Steps to install:

i. Open ***Mozilla Firefox*** Browser
ii. Go to ***Tools*** & select ***Add-ons***
iii. Click on ***Tools button*** & select ***Install Add-on from File***
iv. Browse & select '***Selenium-ide-2.5.0.xpi***' file, click ***Open***, click ***Install Now*** & click Restart Now.

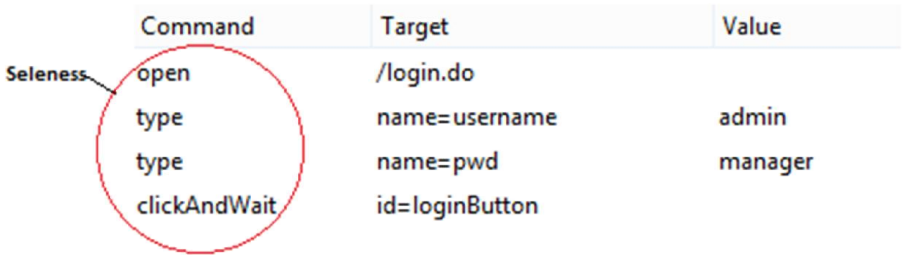
Record & Playback using Selenium IDE:

i. Open the ***Mozilla Firefox browser***, then open the ***Selenium IDE*** which will be in ***Recording mode by default.***
ii. Perform required action on the application such as ***opening the login page, entering Username, entering password, clicking on Login button etc***.
iii. Click on the '***Red Button***' to stop the Recording.
iv. Go to ***File***, select '***Save testcase'*** specify the ***testcase name*** & click ***Save***.
v. In order to playback the recorded steps, click on ***Play current testcase button*** present on the ***Toolbar*** of ***Selenium IDE***.

Note:

The Selenium IDE script will be saved in ***html format***.

a. The Recorded steps will be displayed in table format which contains following 3 columns:
   i. ***Command***: It represents the operation.
   ii. ***Target***: It represents the element on which the operation has to be performed.
   iii. ***Value***: It represents the input required to perform the operation.



Q. What is Seleness?

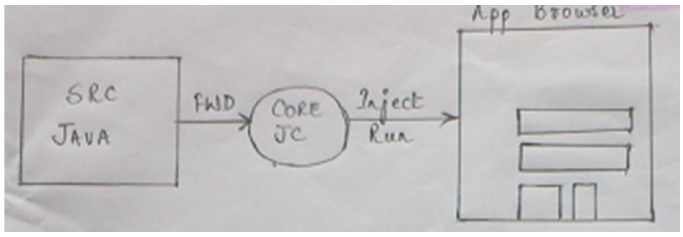
A: The commands of Selenium IDE are called as ***Seleness***, ***such as open, type, click*** etc.

Limitations of Selenium IDE:

i. It supports only Mozilla Firefox browser.
ii. We cannot write any looping statement.
iii. We cannot write any conditional statement.
iv. We cannot take any data from external source such as Excel file, Database etc.
v. Handling multiple browsers is not possible.
vi. Handling pop ups is not possible or very difficult.
Note: To overcome these limitations we can for ***Selenium*** RC.

Selenium RC:

Selenium RC is a wrapper built around Selenium Core. In Selenium RC, we can write the code in different languages such as Java, PHP etc. This will overcome all the limitations of Selenium Core. During runtime 'Selenium RC' will convert the automation script into java script statements & it will forward it to Selenium Core, where Selenium Core will inject that java script into the browser & executes it.



Sample Selenium RC code:

```
import com.thoughtworks.selenium.*;

public class TestCase1 extends SeleneseTestNgHelper
{
    @Test
    public void testcase1()
    {
        selenium.open("login.do");
        selenium.type("name=username","admin");
        selenium.type("name=pwd", "manager");
        selenium.click("id=loginButton");
    }
}
```

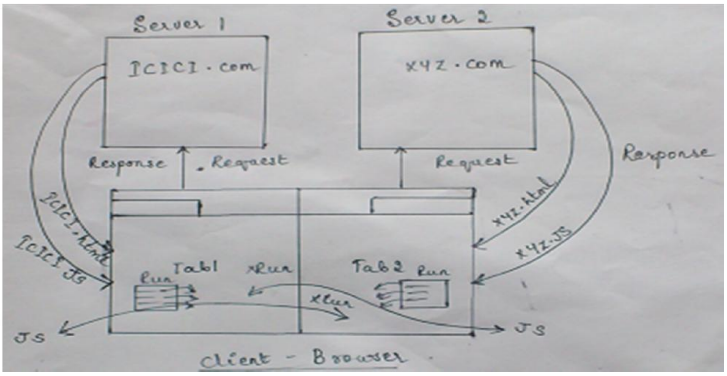
Note:

Selenium RC is better than ***Selenium Core*** but it has ***same origin policy issue***.

Same Origin Policy:

As per ***W3C (World Wide Web Consortium) standards***, browser should allow the script to execute only if the origin of both ***html*** & the ***java script*** are same.

Ex:



Description:

In the client machine the browser is launched with ***2 tabs***, in the ***first tab*** when we type www.icici.com, the request will go ***to icici.com*** server which will respond back with respective ***html & java script***.

In the ***second tab***, when we type www.xyz.com the request will go to www.xyz.com server which will respond with respective ***html & the java script***.

As per the ***same origin policy***, ***Selenium java script*** of ***icici.com*** can be executed only on ***icici webpage***, similarly ***java script of xyz.com*** can be executed only on ***xyz webpage***, ***vise-versa is not allowed***.

Executing ***java script*** of one website on another website is called as ***Cross Site Scripting (XSS)*** & for security reason this will not be allowed by the browser.

In order to execute ***Selenium RC code*** on the web application, we should store the ***Automation script*** in the server machine itself where the web application is installed or we should use proxy server.

In order to overcome this limitation we can use the latest version of selenium which is '***Selenium WebDriver'.***

Selenium WebDriver:

It is a ***web application automation tool*** which performs the operation on the application by using ***native methods*** of the browser.

Configuring WebDriver:

Required software,

1. JDK
2. Eclipse IDE
3. Mozilla Firefox browser
4. Selenium Jar file

Note: OS should be windows7 or windows8 & also ensure that all the above software's are updated to latest version.

Downloading Selenium jar file:

1. Open the browser & go to the following website: www.seleniumhq.org/download

2. In the above webpage, go to ***'Selenium Server'*** section, click on download version 2.39.0, click ***Save File*** on the popup, it will save the file '***selenium-server-standalone-2.39.0.jar'*** in ***Downloads folder*** & it will be approximately 33mb.

3. If the file extension is not displayed in the ***Downloads folder***, Go to ***Start*** & type '***Folder'*** select '***Folder Options'***, go to ***View tab*** uncheck '***Hide extensions for known file types'*** checkbox & click ***Ok***.

4. Create a folder in the required location (***Ex: D:\ drive***) with the name '***Workspace'*** & copy paste the selenium jar file into this folder.

5. Open the ***Eclipse IDE***, go to ***File*** → Switch '***Workspace'*** → Select '***Other'*** browse & select above created folder workspace '***D:\Workspace*** click ***Ok*** which will close & reopens the Eclipse IDE.

6. Go to ***File → New → Project***, select ***java project***, click ***Next***, specify the ***project name*** as '***Automation'***, click ***Finish*** & click ***Yes***

7. Right click on '***src'*** folder, go to ***New*** & select '***Package'***.

8. Specify the ***package name*** as '***com.qspiders'*** it should be in lowercase & click ***Finish***.

9. Right click on the ***package***, go to ***New***, select ***Class***, specify the class name (Ex: ***Demo***), select ***PSVM checkbox*** & click ***Finish***.

10. Write a code to print some text using ***SOP ();*** execute & ensure that it is working fine.

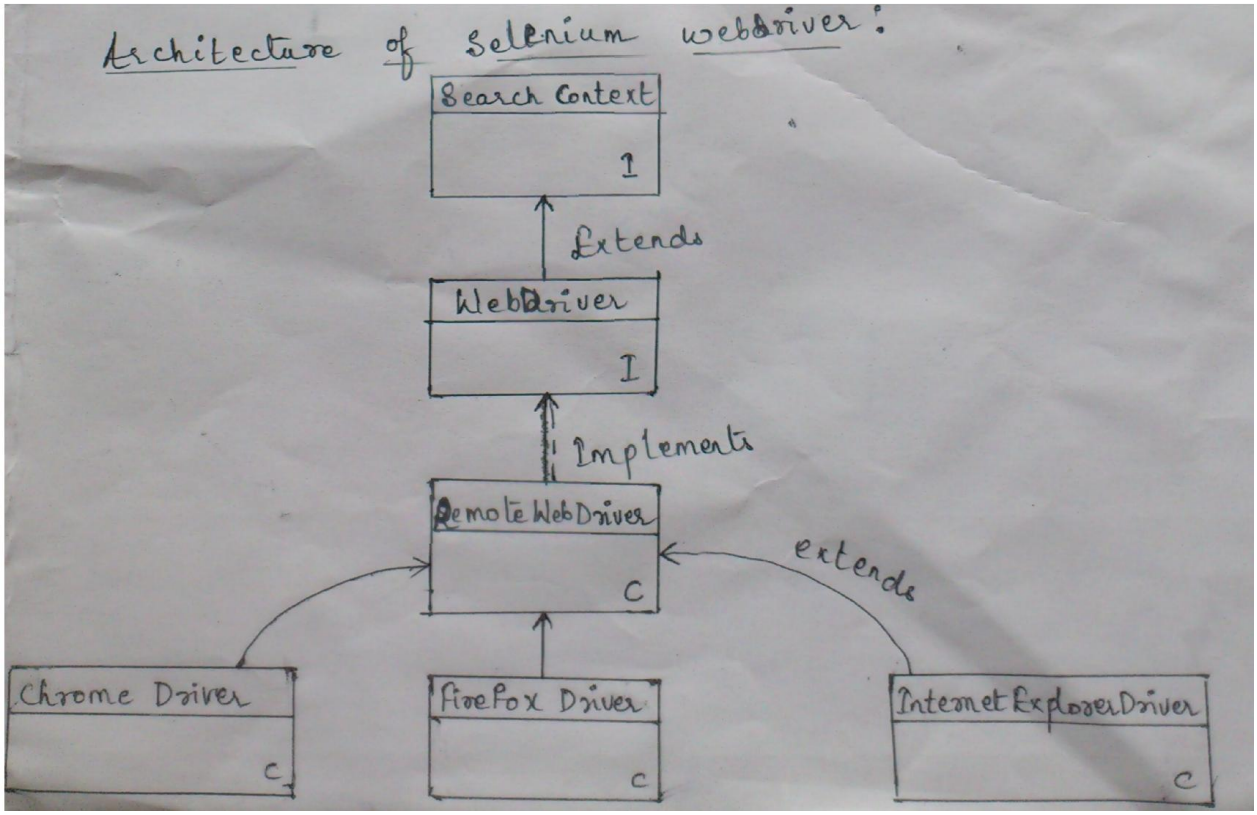
11. Right click on the ***Java Project***, go to ***Build path***, select '***Add external archives***', navigate to the location where selenium jar file is present, select it, and click open which will associate the selenium jar file with the Java Project.

Write the following code:

```
package com.qspiders;
import org.openqa.selenium.firefox.FirefoxDriver;

public class Demo
{
    public static void main(String[] args)
    {
    FirefoxDriver driver=new FirefoxDriver();
    }
}
```

The above code will launch Mozilla Firefox browser.

Architecture of Selenium WebDriver:



Generally when we write a code in selenium we should ensure that the same code works on different browser. This can be done using runtime polymorphism concept.

In order to do this while writing the code first we use "Upcasting" & later we use "Downcasting" concept.

Ex:

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

Note:

Whenever we open the browser using selenium, then it always opens the browser with default profile i.e.

i. No Add-ons
ii. ii. No History
iii. iii. No Auto complete
iv. iv. No cookies

This is similar to opening the Mozilla Firefox browser for the first time.

Selenium cannot perform action on already existing browser, each time it will open new browser and performs the operation.

If the Firefox browser is opened by selenium then it will display "WebDriver" in the status bar (right button corner) of Mozilla Firefox. If any script is getting executed then it will be in "Red color" or else it will be in "black color".

Q. How do you open the login page in Selenium?

A:

```
public class Demo
{
 public static void main(String []args)
     {
        Webdriver driver=new FirefoxDriver();
        driver.get("http://demo.actitime.com");
     }
}
```

Q. How do you open the login page without using get method?

A:

```
driver.navigate().to("http://demo.actitime.com");
```

Q. How do you "Refresh" the webpage?

```
driver.navigate().refresh();
```

Q. How do you click *"Back" & "Forward"* using selenium?

```
public class Demo
{
 public static void main(String []args)
        {

            WebDriver driver=new FirefoxDriver();
            driver.get("http://demo.actitime.com"); //get title of page1
            String t1=driver.getTitle();
            System.out.println(t1);
            driver.get("http://demo.vtiger.com");
            String t2=driver.getTitle();//get title of page2
            System.out.println(t2);
            driver.navigate().back(); //click backward
            driver.navigate().forward(); //click forward
         }
}
```

Q. What is the difference b/w close () & quit ()?

A: "**close ()**" will close only the current browser (main browser) where as "**quit ()**" will close all the browsers of current ***WebDriver*** instance.

Ex-1:

```
public class Demo
{
 public static void main(String []args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.get("http://demo.actitime.com");
    driver.close();
    }
}
```

Ex-2:

```
public class Demo
{
 public static void main(String []args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.get("http://demo.actitime.com");
    driver.quit();
    }
}
```

Note: Using selenium we cannot close the browser which is opened manually or the browsers which are opened by the selenium in the previous executions.

Q. How do you close the browser without using close ()?

A: Using “quit ()” method.

Q. How do you maximize the browser?

```
WebDriver driver=new FirefoxDriver();
driver.manage().window().maximize();
```

WebElement: Anything present on the webpage is "***WebElement"***. Such as textbox, button, radio button, dropdown list, error messages etc.

Before performing the operation such as typing, clicking etc., first we should make the selenium to identify the web element uniquely. In order to do this we use **"Locators"** which is a property value or expression of the element. In order to specify the property value or the expression we should have a basic knowledge of HTML.

HTML: HTML stands for Hyper Text Markup Language which is basically used to create webpage documents. We can also create GUI objects (Web Elements) using HTML DOM (Document Object Model).

Writing HTML Code:-

- Open the Notepad
- Write the following code & save it as mypage.html on the desktop.

```
<html>
 <body>
    welcome to HTML
 </body>
</html>
```

- Double click mypage.html, which will open the webpage in the browser.

Note: While writing the HTML code, use predefined keywords of HTML within angle brackets (<>) which are called as HTML tags.

For starting tag there will be an ending tag indicated by forward slash (/). HTML tags are not case sensitive & ending every tag is not mandatory.

Important HTML Codes:

Text field:

```
<input type="text">
```

Password Field:

```
<input type="password">
```

Checkbox:

```
<input type="checkbox">Remember
```

Button:

```
<input type="button" value="Ok">
```

Radio Button:

```
<input type="radio" name="gender">Male
<input type="radio" name="gender">Female
```

File:

```
<input type="file">
```

Address Bar:

```
<textarea></textarea>
```

Dropdown list(for single select dropdown list):

```
Order the items: <br>
<Select name="IT Mart">
<option>Printer</option>
<option>Desktop</option>
<option>Laptop</option>
<option>HardDisk</option>
</select>
```

Multiselect list box:

```
  order the items: <br>
  <Select name="IT Mart" multiple>
  <option>Printer</option>
  <option>Desktop</option>
  <option>Laptop</option>
  <option>HardDisk</option>
  </select>
```

Link or Hyperlink:

```
<a href="http://demo.qspiders.com" title="Click Here">Qspiders</a>
```

Image:

```
<img src="logo.png>
```

Image Link:

```
<a href="http://www.google.com" title="search"><img src="logo.png"></a>
```

Web Table:

```
<table border ="1">
 <tr>
        <th>SlNo</th>
        <th>Name</th>
        <th>Select</th>
</tr>
 <tr>
        <td>1</td>
        <td>Arun</th>
        <td><input type="checkbox"></td>
</tr>
 <tr>
        <td>2</td>
        <td>Bhanu</th>
        <td><input type="checkbox"></td>
</tr>
```

WebElement Programmatically:

```
<a href="URL"abc</a>
```

In Selenium “WebElements” means any element present on the webpage which is created using HTML tag.

The WebElement generally contains 3 components.

1. **HTML Tag:** Any word which is present immediately after the less than symbol (<).

```
Ex:
<html>
<title>
<body>
<input>
<textarea>
<Select>
<option>
<a>
<img>
<table>
<tr>
<td>
<th>
<br> etc.
```

2. Attributes: Any word or pair of words separated by "=" which are present after the "html tag" till the greaterthan symbol(>) are called as Attributes also caled as Property name & Property value.

```
Ex:
type="text"
type="password"
type="checkbox"
type="button"
type=radio"
type="file"
value="Ok"
name="gender"
title="Click Here"
href="http://www.google.com"
src="logo.png"
border="1"
multiple
```

3. Text: Any word which is resent after the greater than(>) symbol of the "html tag" till the end of the respective html tag are called as text of the element.

```
Ex:
>Qspiders<
>Welcome to HTML<
>Male<
>Remember<
>Laptop<
>SlNo<
>Arun<
```

Locators: Selenium will identify the element using the locators.

In selenium we have following 8 types of locators:

```
1.iD
2.name
3.className
4.linkTest
5.partialLinkTest
6.tagName
7.cssSelector
8.xpath
```

All the above locators are available under "By" class of selenium. All of those are static methods, all of them will take an argument of type "String", and all of them will return an object of "By".

We use "findElement" method to search the element present on the webpage based on the specified locator & it returns an object of type "WebElement".

```
Ex-1:
<html>
  <body>
        <a href="http://www.qspiders.com"
        name="a1" class="c1" id="a1">qspiders</a>
 </body>
</html>
```

Write the following code in the java class:

```
class Demo
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.get("file:///C:/Users/Admin/Desktop/mypage.html");
        By b=By.id("a1");
        WebElement e=driver.findElement(b);
        e.click();
        //driver.findElement(b).click()
        driver.findElement(By.id("a1").click(); //Optimised code
    }
}
```

In selenium we always write the optimized code instead of detailed statements.

Clicking on the link present on the above webpage using different locators:

```
1. id()
driver.findElement(By.id("a1").click();

2. name()
driver.findElement(By.name("n1").click();

3. classname()
driver.findElement(By.className("c1").click();

4. linkTest()
driver.findElement(By.linkTest("qspiders")).click();

5. partialLinkTest()
driver.findElement(By.partialLinkTest("spider")).click();

Note: Linktext & partialLinkText can be used only if the element
is a link(html tag should be "<a").


6. tagName()
driver.findElement(By.tagName("a")).click();
```

Firebug:

While writing the code in selenium we need to see the source code of the element, if we select "view page source" then the browser will display source code of complete webpage.

In order to view the source code of required element we use a Mozilla Firefox Add-on it is "Firebug".

Installing firebug:

i. Open Mozilla Firefox browser, go to tools, and select Add-ons.
ii. Type "firebug" in the search field hit Enter.
iii. Click install button of firebug.
iv. Close & reopen the browser.
v. Go to required webpage
vi. Right click on the required element & select "inspect element with firebug" which will display firebug window & it will highlight the source code of selected element.

```
Note: id, name, linkText, xpath(preference order)
```

Login code to login Actitime application:-

```
class Demo
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.get("http://demo.actitime.com");
        driver.findElement(By.name("username")).sendKeys("admin");
        driver.findElement(By.name("pwd")).sendKeys("manager");
        driver.findElement(By.id("loginButton")).click();

    }
}
```

Synchronization:

During runtime when selenium starts performing the operation on the application first it will try to locate the element using the specified locator, if the element is found then it will perform the operation, if the element is not found then it will throw "no such element exception".

Most of the applications are very slow compare to the execution speed of selenium because of this reason the automation script will fail. In order to overcome this we should match the execution speed of selenium with the application which is called as "Synchronization".

We can synchronize the script using sleep() method of thread class as show below.

```
try
    {
    Thread.sleep(1000);
    }
    catch(InterruptedException e)
    {
    System.out.println("can't sleep");
    }
```

The above code will make the script to always wait for 10 seconds.

Limitation of sleep ():

1. It always makes the script to wait for specified amount of duration even though the element is displayed within the deviation.

2. Whenever the application is slow in all such locations, we should use “sleep ()” which will drastically increase total length of the script.

In order to overcome this we can go for "***implicit wait***".

IMPLICIT WAIT:

The actual name of the method is "implicitlyWait" which takes 2 arguments, first one is duration & second one is a Unit such as SECONDS, HOURS, and DAYS etc.

When we use ***"implicitlyWait***" then automatically all the "***findElement ()"*** methods will wait up to the specified timeout.

Ex: If the timeout is 10 sec then during runtime if the element is not located then the selenium will keep searching for the element if it is found at 5th sec it immediately performs the operation & it will not wait till the 10 seconds. If the element is not found even after 10 seconds then only it will through the exception.

Ex: **Selenium script to Login & Logout from ActiTime application.**

```
class Demo
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://demo.actitime.com");
        driver.findElement(By.name("username")).sendKeys("admin");
        driver.findElement(By.name("pwd")).sendKeys("manager");
        driver.findElement(By.id("loginButton")).click();
        driver.findElement(By.id("logoutLink")).click();
        driver.close();

    }
}
```

LOCATING THE ELEMENT:

1. If the specified locator is not matching with any of the element present on the webpage then “findElement ()” method will throw following exception.

"NoSuchElementException"

2. If the specified locator is matching with only one element then the “findElement ()” method returns that WebElement.

3. If the specified locator is matching with multiple elements (Duplicates) then “findElement ()” method returns first WebElement.

cssSelector:

css stands for ***"cascaded style sheathing"*** which is a technology used by the developer to handle presentation layer of the application(to make the look & feel of the application better).

While writing the css code to identify the elements, developer uses an expression which is called as "cssSelector" we can use the same expression in selenium also.

The syntax is

```
htmltag[PropertyName='value']
```

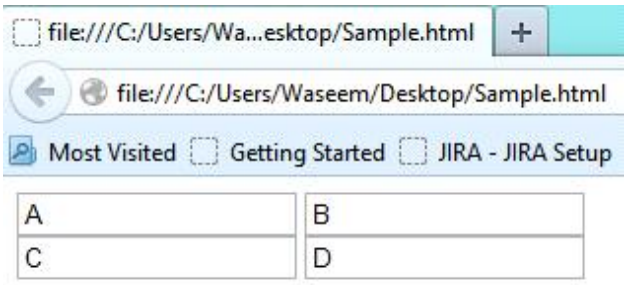
```
Ex:
<html>
 <body>
        UN: <input type="text"><br>
        PW: <input type="password">
 </body>
</html>

class Demo
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(2,TimeUnit.SECONDS);
        driver.get("file:///C:/Users/Waseem/Desktop/mypage.html");
        driver.findElement(By.cssSelect("input[type='text']")).sendKeys("admin");
        driver.findElement(By.cssSelector("input[type='password']")).sendKeys("manager");
        driver.close();
    }

}
```

xpath: xpath is an expression in order to specify xpath expression we start with forward slash(/) which represent the beginning of the html tree which is called as root. After every forward slash (/) we should mention the respective html tag.

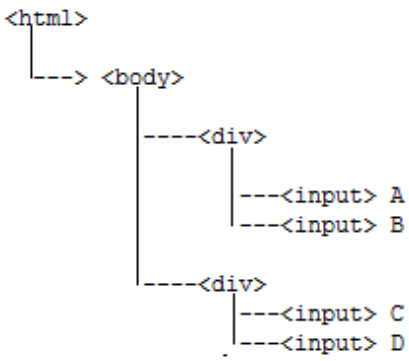
In xpath we can also use the index which **starts from '1'**.

Ex: Sample HTML:

```
<html>
<body>
        <div>
        <input type="text" value="A">
        <input type="text" value="B">
        </div>
        <div>
        <input type="text" value="C">
        <input type="text" value="D">
        </div>
 </body>
</html>
```



HTML Tree:



xpath	Matching Element
1. /html/body/div/input	ABCD
2. /html/body/div[1]/input	AB
3. /html/body/div[1]/input[1]	A
4. /html/body/div[2]/input	CD
5. /html/body/div[2]/input[2]	D
6. /html/body/div/input[1]	AC
7. /html/body/div/input[2]	BD

Firepath:

In order to check whether the xpath expression is correct or not we use "***Firepath***" add-on.

In order to install "***Firepath add-on***" go to "***Tools***", Add-ons and search for "***firepath***" & click ***install button of firepath***, after the installation, close & reopen the Mozilla Firefox browser.

Right click anywhere on the webpage, select "***Inspect element with firebug***", go to "***firepath***" tab type the xpath expression & press Enter.

If the xpath expression is correct, it will highlight the matching element on the webpage in the source code & it will also display the number of matching nodes in the status bar of firebug.

Absolute xpath: Specifying the complete path of the element from the root node (/html) is called as ***"Absolute path".***

Specifying the Absolute path for the element will be very lengthy & time consuming, because of this reason we use "***Relative xpath***".

Relative xpath:

While specifying xpath expression instead of using single forward slash(/) we use double forward slash(//). Using double forward slash (//) we can access any child (any descendants).

Ex:

Xpath	Matching Elements
//input	ABCD
//div[1]/input	AB
//div[1]/input[1]	A
//div[2]/input	CD
//div[2]/input[2]	D
//div/input[1] or //input[1]	AC
//div/input[2] or //input[2]	BD

****xpath by Attribute:**

Relative xpath will reduce the length of expression but we should know the exact index of the element to identify the required element.

Instead of using this "Relative xpath, we use "xpath by attribute" which has following syntax,

```
//htmlTag[@PropertyName='value']
```

```
Ex: 1. //input[@name='username']
    2. //input[@name='pwd']
    3. //input[@name='remember']
```

Note: If the property value is very lengthy in such cases we can use "contains" function of xpath which takes property name & the value as argument.

If the specified value is present inside the property value(substring) then it will identify the element.

```
Ex: 1. //input[@name='username']
    2. //input[@name='pwd']
    3. //input[@name='remember']
```

```
Ex: 1. //input[@id="keepLoggedInCheckBox']
       //input[contains(@id,'LoggedIn']

Ex: 2. //a[@id="loginButton']
       //a[contains(@id,'login')]

Ex: 3. //a[@id='toPasswordRecoveryPageLink']
       //a[contains(id,'Recovery')]
```

Text Function: In order to identify the elements based on its text we should use "Text Function" of xpath which has the following syntax.

```
//htmlTag[text()='value']
```

```
Ex: 1. //div[text()='Tasks']
    2. //a[text()='Login']
    3. //a[text()'Forgot your password?']
```

Q. In the login page *"Forgot password link"* is keep changing i.e. sometimes it will be *"Did you Forgot the password"* sometimes it will be *"Forgot your password"* how do you identify this dynamically changing elements.

```
A: //a[contains(text(),'Forgot')]
```

Q. How do you identify the dynamically changing inbox link.

```
A: //a[contains(text(),'Inbox')]
```

Limitations of xpath: If there are additional spaces before or after the string in the property value then xpath will not identify the element, in such cases we should use "contains function".

```
Ex: <a> Nokia 108 (Red) </a> ==> code
    1. //a[text()=' Nokia 108 (Red) '] --> will not work
    2. //a[contains(text),'Nokia 108 (Red)')] --> will work
```

Advantages of contains() function or when we should use contains function:

We use contains function if,

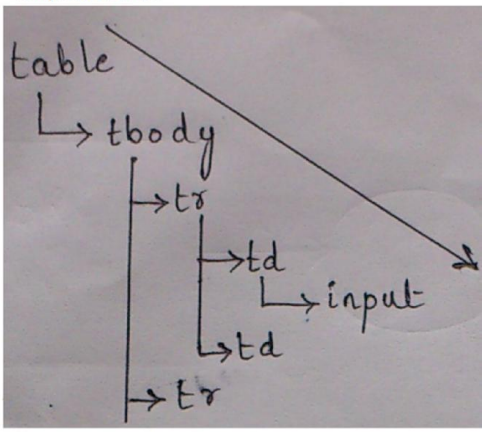
i. If the value is very lengthy.
ii. If value is keeps changing.
iii. If value has additional spaces.

Traversing in HTML Tree using xpath:

We can derive the xpath expression using which we can perform navigation in the html tree i.e. we can travel in the forward direction & we can also travel in the backward direction. We navigate in the backward direction to find the parent element where as we travel in the forward direction to find the child element.

Traversing in Forward direction:

Sample HTML Tree:



//table/tbody/tr[1]/td[1]/input

Traversing in backward / upward direction:

Syntax:

```
//parent htmltag[childxpath]
```

```
Ex: //input[@type='text']
    //td[input[@type='text']]
    //tr[td[input[@type='text']]]
    //tbody[tr[td[input[@type='text']]]]
    //table[tbody[tr[td[input[@type='text']]]]]
```

Identifying the Dependent elements:

Sometimes we need to identify an element based on some other elements, such type of element are called as ***"Dependent elements".***

In order to identify the dependent element we should follow below mentioned steps.

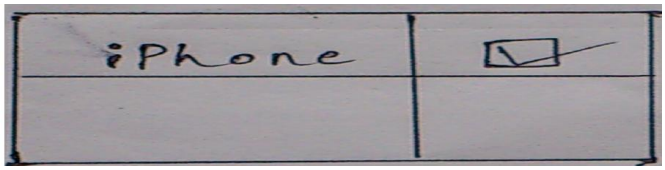
Examples:

1. Derive the x-path for identifying the *checkbox of iPhone*.

Step 1: For the given requirement identify the dependent & independent element.

Checkbox is Dependent element

IPhone is Independent element



Step 2: Inspect the independent element.

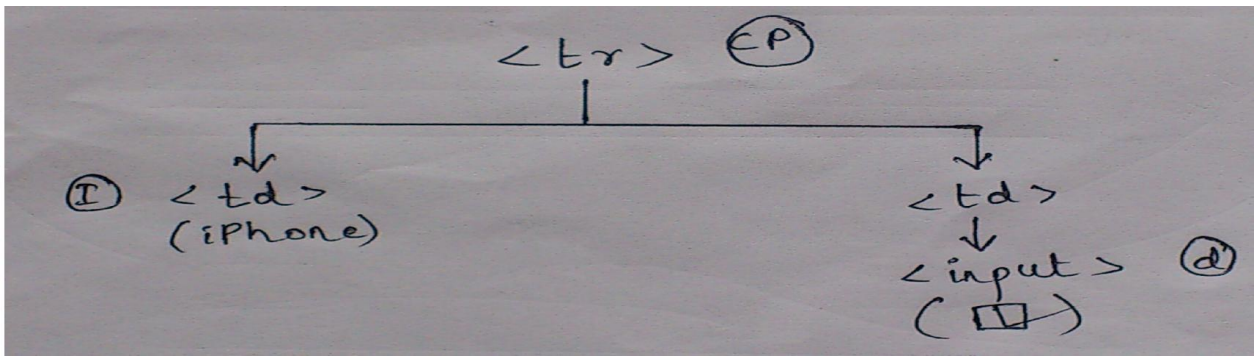
Ex: Right click on IPhone which is present inside the web table & select “Inspect Element with firebug”

Step 3: Identify the common parent (the node which is common for both independent & dependent element in html tree).

Ex: Place the mouse pointer on the source code of independent element in the html tree of firebug & move the mouse pointer in the upward direction till both independent & dependent elements are highlighted.

```
tr is the common parent
```

Step 4: Write the tree structure which contains independent, dependent & common parent.



Step 5: Derive the xpath for independent element.

```
Ex: //td[text()='iPhone'] ==> Independent
```

Step 6: Derive the xpath for independent element.

```
Ex: //tr[td[text()='iPhone']] ==> Common Parent
```

Step 7: Derive the xpath for dependent element.

```
Ex: //tr[td[text()='iPhone']]/td[2]/input ==> Dependent
```

2. Derive the xpath for identifying checkbox of IBM server which is present in “Products” table of vtiger.com

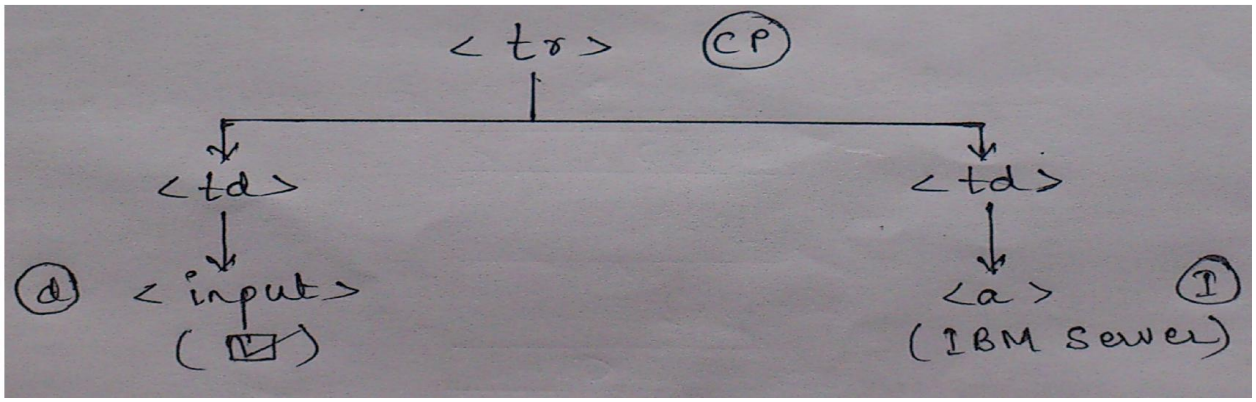
Step 1: Dependent Element is ‘checkbox’

Independent Element is ‘IBM server’

Step 2: Right click on IBM server & select inspect element with firebug.

Step 3: Place the mouse pointer on source code of IBM server in the html tree & move the mouse pointer in upward direction till the checkbox & IBM server is highlighted. In this example common parent is <tr>

Step 4: Write the tree structure.



Step 5: xpath for independent element

```
//a[text()='IBM SERVER']
```

Step 6: xpath for common parent

```
//tr[td[a[text()='IBM SERVER']]]
```

Step 7: xpath for dependent element

```
//tr[td[a[text()='IBM server']]]/td[1]/input
```

3. Derive the xpath to identify the helpline number of Mumbai in www.irctc.com

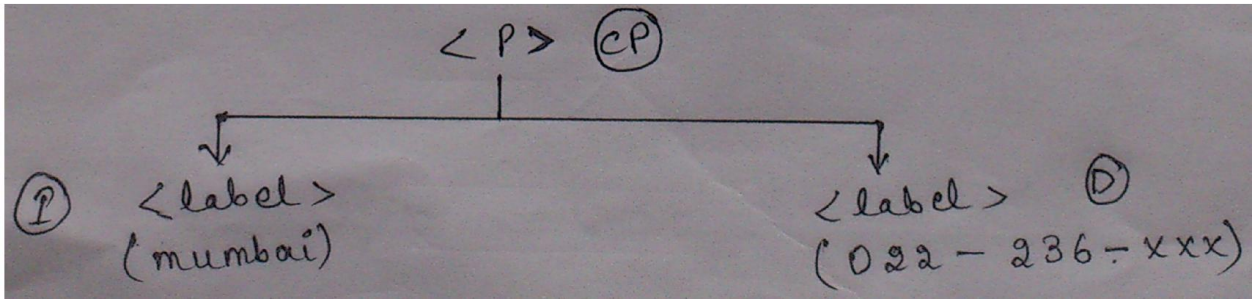
Step 1: Independent is Mumbai & Dependent is 'helpline number'.

Step 2: Right click on Mumbai & click on Inspect element with firebug.

Step 3: Find the Common Parent i.e. place the mouse pointer on the source code of Mumbai & move it in the upward direction.

In this example Common Parent is '<p>'

Step 4: Tree structure



Step 5: xpath for independent element

```
//label[text()='Mumbai']
```

Step 6: xpath for common parent

```
//p[label[text()='Mumbai']]
```

Step 7: xpath for dependent element

```
//p[label[text()='Mumbai']]/label[2]
```

4. Derive the xpath to identify "Add to compare checkbox of Nokia 108 Black mobile phone in flipkart.com

Step 1: Here

Independent element is – **Nokia 108 Black**

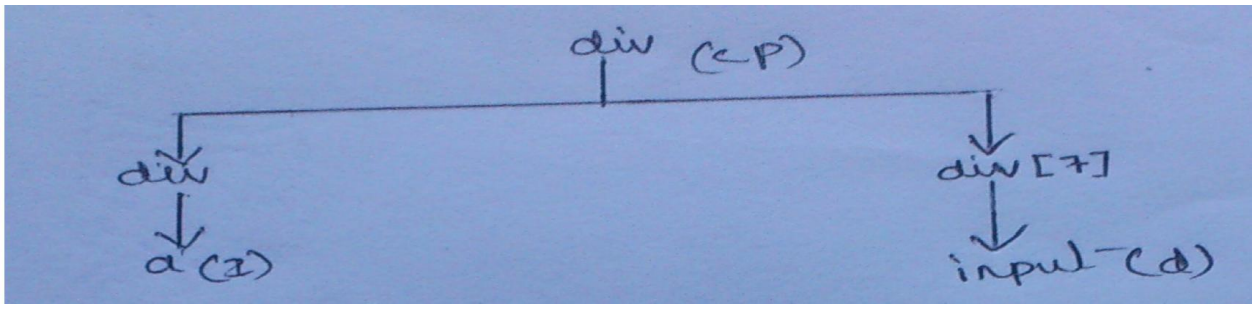
Here Dependent element is **checkbox.**

Step 2: Right click **on Nokia 108 Black** & click on Inspect element with firebug.

Step 3: Common parent is

```
Code: <a title= "Nokia 108 (Black)" Nokia 108 (Black) </a>
```

Step 4: Tree structure



Step 5: xpath to find Independent path.

```
//a[@title='Nokia 108 (Black)']
            or
//a[contains(text(),'Nokia 108 (Black)]
```

Step 6: xpath to find Common parent.

```
//div[div[a[@title='Nokia 108(Black)']]]/div[7]/input
```

Step 7: xpath to find Dependent Element

5. Derive the xpath for identifying the cost of Nokia 108 Black mobile phone present on flipkart.com.

Step 1: Independent element – **Nokia 108 Black**

Dependent element - **Cost**

Step 2: Right click on **Nokia 108 Black** & click on Inspect element with firebug.

Step 3: Identify the common parent (The node which is common for both independent & dependent element in html tree.

**" ** is the common parent

Step 4: Tree structure

Code: //a[@title='Nokia108(Black)']

Step 5: xpath for Independent element**.**

```
//a[@title='Nokia 108(Black)']
```

Step 6: xpath for Common Parent**.**

```
div[a[@title='Nokia 108(Black)']]
```

Step 7: xpath for Dependent element**.**

//div[div[a

6. Derive the xpath expression to identify the checkbox of 'Create testcase task' present in open task page of ActiTime application.

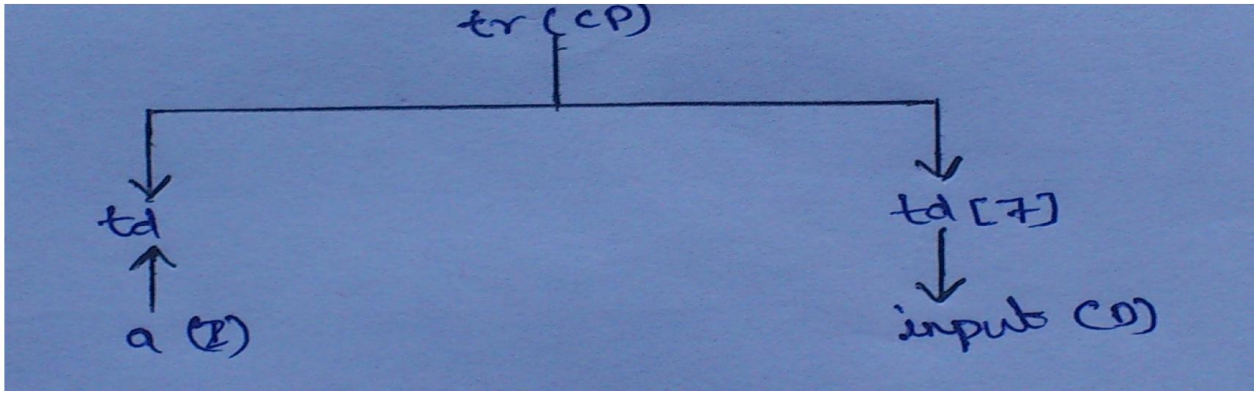
Step 1: Independent element is 'Create test case'

Dependent element is 'checkbox'

Step 2: Inspect Create Testcase link

Step 3: Common parent is <tr>

Step 4: Tree Structure



Step 5: xpath for independent element.

```
//a[text()='Create TestCase']
```

Step 6: xpath for Common Parent

```
tr[td[a[text()='Create TestCase']]... pending
```

Step 7:

```
tr[td[a[text()='Create TestCase']]/td[7]/input
```

7. Derive the xpath for identifying the checkbox of Apple customer present in ActiTime application.

Step 1: Independent element is Apple

Dependent element is Checkbox

Step 2: Inspect the Apple link

Step 3: Common parent is <tr>

Step 4: Tree structure

Step 5: xpath for Independent element

```
//a[text()='Apple']
```

Step 6: xpath for Common parent

```
tr[td[table[tbody[tr[td[a[text()='Apple']]]]]]]
```

Step 7: xpath for dependent element

```
tr[td[table[tbody[tr[td[a[text()='Apple']]]]]]]/td[6]/input
```

Important Locators:

In selenium we have 8 types of locators out of which following 4 are important.

```
1.id
2.name
3.linkText
4.xpath
```

Important methods of WebElement:

i. sendKeys(): This method is used to type the input on textbox password field & text area.

ii. click(): This method is used to click on any element such as button, link, image link, checkbox & radio button.

Q. How do you enter the first name & second name in the different lines of multi textbox.

Ans: we use **"\n"**

```
Ex: driver.findElement(By.id("addr")).sendKeys("Raj\n Kumar");
```

iii. clear(): This method is used to remove the text present in textbox, password field or text area.

Q. How do you replace the text present in the textbox?

A: Fire we will remove the text using clear() & then we type the new text using sendKeys().

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class Demo
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://demo.vtiger.com/");
        driver.findElement(By.id("username")).clear();
        driver.findElement(By.id("username")).sendKeys("manager");
    }
}
```

iv. getAttribute(): This method is used to get the property value of the specified property name.

Q. How do you retrieve the default value present in the text box?

```
Ans: Using "getAttribute()" method

package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class Demo
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://demo.vtiger.com/");
        String s=driver.findElement(By.id("username")).getAttribute("value");
        System.out.println("Default UN:"+s);
```

v. getText(): This method is used to get the text of the element.

Q. Write a code to get helpline number of Mumbai present in irctc.com.

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class Demo
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://irctc.com/diplay.servlet");
        String xpExp="//P[label[text()='Mumbai']]/label[2]";
        String helpNum=driver.findElement(By.xpath(xpExp)).getText();
        System.out.println(helpNum);
    }
}
```

Q. What is the difference b/w getText() & getAttribute().

A: "getText()" is used to get the text of the element, where as "**getAttribute()**" is used to get the property value of the element.

Q. How do you login to the application without clicking on the login button?

A: We can login to the application by pressing Enter Key after typing or entering username & password instead of login button.

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class Demo
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://demo.actitime.com/login.do");
        driver.findElement(By.id("username")).sendKeys("admin");
        driver.findElement(By.name("pwd")).sendKeys("manager");
        driver.findElement(By.name("pwd")).sendKeys(Keys.RETURN);
    }
}
```

Q. Write a code to type user in username field of Acti Time application, copy the user name & paste it into the password field.

isSelected(): This method is used to check whether the specified check box or radio button is selected or not. It returns true if the check box or radio button is selected, it returns false if the check box or radio button is not selected.

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class Demo
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://demo.actitime.com/login.do");
        driver.findElement(By.name("remember")).click();
        boolean selected=driver.findElement(By.name("remember")).isSelected();
        if(selected)
            {
            System.out.println("check box is Selected");
            }
        else
            {
            System.out.println("Check box is not Selected");
            }
    }
}
```

Note:

In WebElement interface we also have "isDisplay()" method to check whether element is displayed not & "isEnabled()" method to check whether the element is enabled or disabled.

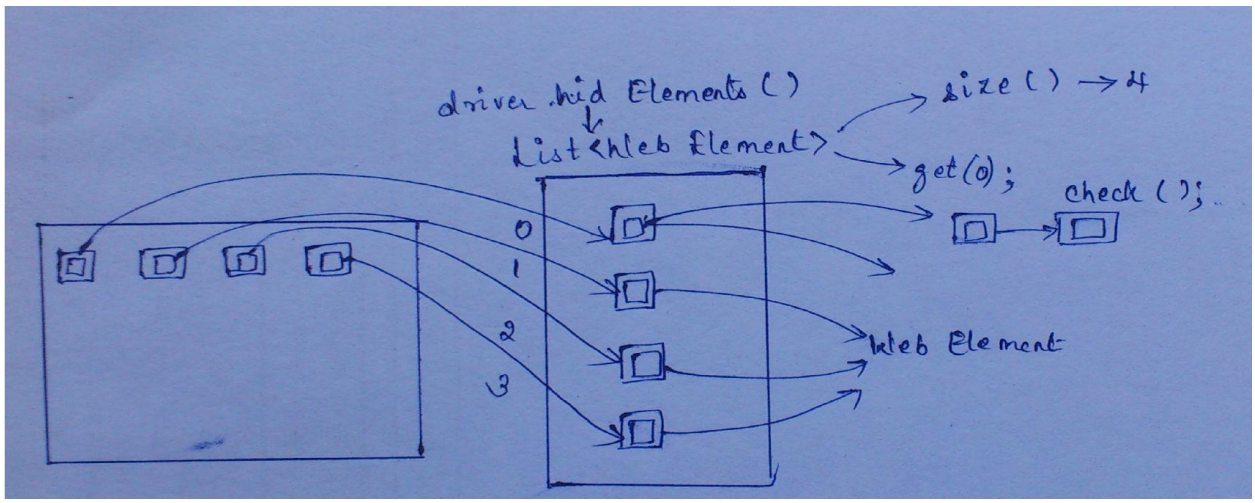
Handling Multiple Elements:

In order to perform operation on multiple elements, we use "findElements()" method which returns list of WebElement("list<WebElements>").

We use **"size ()"** method to count the number of elements present in the list & we use get() to get the WebElement present in the list.

Ex: Counting the number of characters & selecting all the check boxes present on the webpage.

```
package com.qspiders;

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.firefox.FirefoxDriver;

public class LinkCount
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driver.get("http://www.gsmarena.com/nokia-phones-1.php");
        List<WebElement> allCheckBox=
                driver.findElements(By.xpath("//input[@type'CheckBox']"));
        int c=allCheckBox.size();
        System.out.println("CheckBox Count:"+c);
        for(int i=0; i<c; i++)
        {
            WebElement a=allCheckBox.get(i);
        }
    }
}
```



Q. Write a code to count the number of links present on the webpage & also print name of all the links (link Text).

```
package com.qspiders;

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.firefox.FirefoxDriver;

public class LinkCount2
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(15,TimeUnit.SECONDS);
        driver.get("http://www.gsmarena.com/nokia-phones-1.php");
        List<WebElement> allLinks=driver.findElements(By.xpath("//a"));
        int linkCount=allLinks.size();
        System.out.println("linkCount:"+linkCount);
        for(int i=0; i<linkCount; i++)
        {
            String linkText=allLinks.get(i).getText();
            System.out.println(linkText);
        }
        driver.close();
    }
}
```

Q. Write a code to select alternative checkboxes present in gsmarena.com

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

public class CeckBoxes2
{
    public static void main(String[] args)
    {
    WebDriver driver = new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
    driver.get("http://demo.actitime.com/login.do");
    WebElement web=driver.findElement(By.id("username"));
    web.sendKeys("user");
    web.sendKeys(Keys.CONTROL,"a");
    web.sendKeys(Keys.CONTROL,"c");
    driver.findElement(By.name("pwd")).sendKeys(Keys.CONTROL,"v");
    }
}
```

Q. Write a code to select all the check boxes present in gsmarena.com from right to left (reverse order).

Q. Write a code to print name of all the links present on the webpage excluding the links that do not have the text.

Handling the List Box (Dropdown):

Using Web Element we cannot perform the required operation of the list box, in order to handle the list box we should use **"Select"** class.

First we will identify the list box using "**find Element ()"** method then we create instance of "Select" class by specifying "**list Element**" as argument for "**Select"** class constructor, then we call anyone of following method.

1. select.selectByIndex(int);
2. select.selectByvalue(String);
3. select.selectByVisibleText(String);

```
package com.qspiders;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;


public class ListBox2
{
    public static void main(String[] args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(15,  TimeUnit.SECONDS);
    driver.get("http://www.fatcow.com/");
    WebElement cElement=driver.findElement(By.name("country"));
    Select select=new Select(cElement);
    //Select.selectByIndex(1);
    //Select.selectByvalue("CAD");
    select.selectByVisibleText("Australia");
    }
}
```

Handling Multi select List Box:

In order to perform operation on Multi Select List Box we will use “Select” class only & we can also use these select methods on multi select list box.

When we use “Selectby()” method it will keep already selected option as it is & it will also select newly specified option.

Ex: Sample Code:

```
<html>
 <body>
    <select name="country" multiple>
    <option value="USD">United States</option>
    <option value="GBP">United Kingdom</option>
    <option value="CAD">Canada</option>
    <option value="AUD">Australia</option>
    <option value="EUR">Europe</option>
    </select>
 </body>
</html>
```

Ex:

```
package com.qspiders;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

public class MultiList
{
    public static void main(String[] args)
    {
        WebDriver driver = new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driver.get("file:///C:/Users/Waseem/Desktop/Sample%20Code.html");
        WebElement cElement = driver.findElement(By.name("country"));
        Select select=new Select(cElement);

        select.selectByIndex(1);
        select.selectByValue("CAD");
        select.selectByVisibleText("Europe");

        select.deselectByIndex(1);
        select.deselectByValue("CAD");
        select.selectByVisibleText("Europe");
    }
}
```

Q. Write a code to count the number of options present in "Multi-select List box" & select all the options.

```
package com.qspiders;

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.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

public class MultiList2
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://www.fatcow.com/");
        WebElement cElement = driver.findElement(By.name("country"));
        Select select=new Select(cElement);
        List<WebElement> allOptions=select.getOptions();
        int count=allOptions.size();
        System.out.println(count);
        for(int i=0; i<count; i++)
        {
            select.selectByIndex(i);
        }
        select.deselectAll();
    }
}
```

Note:

In **"select"** class we have a method called ***"deselectAll()"*** but there is no method called ***"selectAll"***. In order to select all the options present in the list box, we should use looping statement as show above.

Q. How do you check whether the List box is dropdown list or Multi-select list.

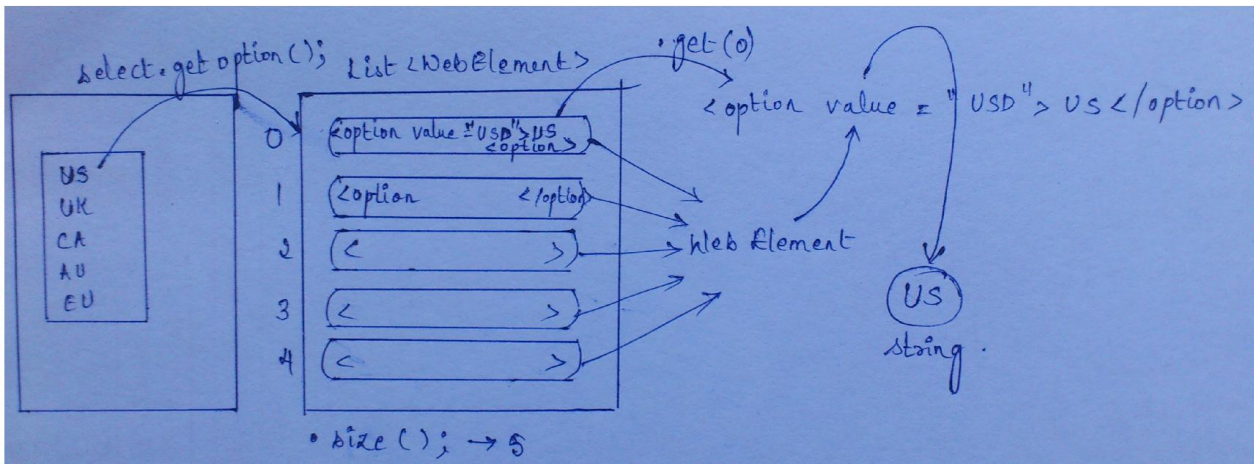
A: using "isMultile();"

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

public class DropdownOrMultiselect
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://www.fatcow.com/");
        WebElement cElement = driver.findElement(By.name("country"));
        Select select=new Select(cElement);
        if(select.isMultiple())
        {
            System.out.println("It is Multiselect Listbox");
        }
        else
        {
            System.out.println("It is Dropdown Listbox");
        }
    }
}
```

Q. Write a code to print all the options present in the List Box.

```
package com.qspiders;

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.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

public class ListBox3
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://www.fatcow.com/");
        WebElement cElement = driver.findElement(By.name("country"));
        Select select=new Select(cElement);
        List<WebElement> allOptions=select.getOptions();
        for(int i=0; i<allOptions.size(); i++)
        {
            WebElement a =allOptions.get(i);
            String t=a.getText();
            System.out.println(t);
        }
    }
}
```

```
Output:
United States
United Kingdom
Canada
Australia
Europe
```



Q. Write a code to search for the specified option present in the List Box.

```
package com.qspiders;
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.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

public class SearchList
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://www.fatcow.com/");
        WebElement cElement = driver.findElement(By.name("country"));
        Select select=new Select(cElement);
        List<WebElement> allOptions=select.getOptions();
        String ev="Canada";
        String msg="Not Found";
        for(int i=0; i<allOptions.size(); i++)
        {
            String av=allOptions.get(i).getText();
            if(ev.equals(av))
            {
                msg="Found";
                break;
            }
        }
        System.out.println(msg);
    }
}
```

Q. How do you select an option present in Multi-select List box without using "select" class.

A: In order to select the option present in multi-select list box we can use x-path to identify the option & we use "click ()" method to select the option.

```
Ex: String = x="//select[@name='country']/option[3]";
    driver.findElement(By.xpath(x)).click();
    // //select[@name='country']/option[text()='canada']
```

Assignment:

1. **Write a code to print all the options present in the list box in Reverse order.**
2. **Write a code to select alternative options present in the Multi-select List box.**
3. **Write a code to check whether the specified option is Duplicate or not.**

Handling custom list box:

In order to select required option present in the list box, we can use list class only if the list box is created using **<Select> html tag** then it is called as "**custom list box**" where we cannot use the "**Select class**" of selenium.

To handle custom list box we use **"sendKeys()"** method only, first we type the name of the required option then we press enter key, if required we can also press down arrow key required number of times.

```java
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class ListBox1
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://www.yatra.com/");
        driver.findElement(By.id("BE_Flight_origin_city")).sendKeys("goa");
        driver.findElement(By.id("BE_flight_origin_city")).sendKeys("RETURN");

    }

}
```

Handling Dropdown Menu:

In order to handle "**Dropdown Menu**" i.e. to move the mouse pointer on the menu so that we select the submenus we use "**Actions**" class of selenium.

In order to use "**Action**" class first we need to create instance of "**Actions**" class by specifying "driver" as the argument for "**Actions**" class constructor.

Whenever we call any method of "Actions" class we must call "**perform()**" method.

Ex: Selecting "**Contact Us**" submenu which is present under "**Customer Support**" Menu of http://www.yatra.com/.

```java
package com.qspiders;

import java.util.concurrent.TimeUnit;

public class DropdownMenu
{
    public static void main(String[] args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
    driver.get("http://www.yatra.com/");
    WebElement menu = driver.findElement(By.partialLinkText("Customer Support"));
    Actions actions=new Actions(driver);
    //move mouse on "customer support" menu
    actions.moveToElement(menu).perform();
    //select "Contact Us" submenu
    driver.findElement(By.linkText("Contact us")).click();
    }

}
```

Q. Write a code to open the link in *new tab*.

```
package com.qspiders;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.interactions.Actions;

public class NewTab
{
    public static void main(String[] args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
    driver.get("http://www.yatra.com/");
    WebElement supportLink = driver.findElement(By.partialLinkText("Customer Support"));
    Actions actions=new Actions(driver);
    //Right click on Customer support link
    actions.contextClick(supportLink).perform();
    //press 't' to open the link in NewTab('w' for New Window)
    actions.sendKeys("t").perform();
    }
}
```

Summary:

i. We use "**Select**" class to handle dropdown list & multi select list box.
ii. We use "**sendKeys()**" method to handle custom list box.
iii. We use **";Actions"** class to handle dropdown menu & Context menu.

Handling Pop ups:

In selenium code to handle the pop up depends on the type of the popup.

The pop up can be categorized as

i. Alert & Confirmation Popup
ii. Hidden Division Popup
iii. Page on load Popup
iv. File Upload Popup
v. File Download Popup
vi. Child Browser Popup
vii. Window Popup

Important Note:

In order to recognize the pop up type always use the browser which opened by selenium webdriver.

Alert & Confirmation Popup:

Characteristics:

i. We can move these pop ups.
ii. We cannot inspect the pop up.
iii. It will have "**OK**" button (**Alert**) or "**OK**" "**Cancel**" button (**Confirmation pop up**).

Solutions:

In order to handle alert pop up first we should transfer the control from webpage to the alert pop up, then we can use "**getText()**" method to get the message displayed on the alert pop.

In order to click on "**OK**" button we use "**accept()**" method.

In order to click on Cancel button we use "**dismiss()**" method of "**alert class**".

Once the **alert pop up** is closed, the control will be automatically transferred back to the webpage.

```
package com.qspiders;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class AlertConfirmation
{
    public static void main(String[] args)
    {
        WebDriver driver = new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        //login to vtiger.com
        driver.get("http://demo.vtiger.com/");
        driver.findElement(By.xpath("//button[text()='Sign in']")).click();
        //Clicks on Products
        driver.findElement(By.linkText("Products")).click();
        //Click on Action & select Delete
        driver.findElement(By.xpath("/strong[text()='Actios']")).click();
        driver.findElement(By.linkText("Delete")).click();
        //switch to alert pop up
        Alert alert=driver.switchTo().alert();
        //Get the message & print it
        String msg=alert.getText();
        System.out.println(msg);
        //Click on 'OK' button
        alert.accept();
        //to click Cancel use --> alert.dismiss()
    }
}
```

Write code to perform following steps:

i. Login to ActiTime application, go to "**Tasks**" page then go to "**Project & Customers**"
ii. Click on "**Create customer**" enter customer name & click on "**Cancel**" which will display confirmation pop up. Print the message displayed on the confirmation pop up & click on **Cancel** button which is present on the Confirmation pop.

Q. Write the code to verify whether the alert pop up is displayed or not.

```
try
{
     Alert alert=driver.switchTo().alert();
     String msg=alert.getText();
     System.out.println("Alert is present its msg is :"+msg);
     alert.accept();
}
catch(NoAlertPresentException e)
{
     System.out.println("Alert is not present");
}
```

Hidden Division pop up:

Characteristics:

i. We cannot move the pop up.
ii. We can inspect the pop up (it will be colorful).

Note:

This Popup is part of webpage only which is created using html tag, initially the value of the display property will be "none" (hidden). In order to display this pop up it will be changed to ("Block").

Solution:

Since we can inspect it & it is part of html code we handle it using "findElement()" method only.

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class HiddenDiv
{
     public static void main(String[] args)
     {
     WebDriver driver = new FirefoxDriver();
     driver.manage().timeouts().implicitlyWait(59,TimeUnit.SECONDS);
     //login to actitime
     driver.get("http://demo.actitime.com");
     driver.findElement(By.name("username")).sendKeys("admin");
     driver.findElement(By.name("pwd")).sendKeys("manager");
     driver.findElement(By.id("loginButton")).click();
     //Go to tasks page & select all tasks, Click delete
     driver.findElement(By.xpath("//div[text()='Tasks']")).click();
     driver.findElement(By.linkText("All")).click();
     driver.findElement(By.xpath("//input[@value='Delete Selected Tasks']")).click();
     //Get the message present on Hidden division pop up
     String msg = driver.findElement(By.id("deleteDialogDiv")).getText();
     System.out.println(msg);
     //click delete button present on Hidden division pop up.
     driver.findElement(By.id("deleteButton")).click();
     }
}
```

Calendar Popup:

Calendar pop up is also a type of hidden division pop up & to select the date which is present on the calendar pop up, we use “findElement()” method itself.

Ex: Selecting departure date in yatra.com

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class Calendar1
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("http://www.yatra.com");
        driver.findElement(By.id("BE_flight_depart_date")).click();
        driver.findElement(By.id("a_2014_2_19")).click();
    }
}
```

Assignment:

Q. Write a code to select today’s date in the Calendar.

Q. Write a code to select next month same date in the Calendar.

File Upload popup:

Characteristics:

1. There will be a browse button, clicking on which will display a pop up with the name "File upload".

2. When we inspect the browse button the html tag will be "input" & the type will be "File".

Solutions:

To handle file upload popup we use "sendKeys()" method where we specify complete path of the file & we must use double backward slash(\\) & not the forward slash(/).

Path is not care sensitive & we cannot use the "Relative Path".

```
Ex: uploading the file "toshared.com".
```

Code:

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class FileUpload
{
    public static void main(String[] args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("http://www.2shared.com/");
    driver.findElement(By.id("upField")).sendKeys("d:\\my.cv.docx");
    driver.findElement(By.xpath("//input[@title='Upload File']")).click();
    }
}
```

Note:

If the specified file does not exist or if the size of the file is more than the limit or if it is "**exe**" or "**script file**" the application may through the error message we need to handle it separately.

File Download Popup:

Characteristics:

1. We can move the pop up.

2. We cannot inspect the pop up.

3. It will have 2 radio buttons, "open with" & "save file" and it will have a checkbox.

Solution:

In selenium we cannot perform operation on File Download pop up. In order to handle this file Download pop up we change the settings of the browser so that the file gets downloaded automatically without displaying ***"File Download Pop up"***.

In order to see the settings upon Mozilla Firefox browser, type "***about config***" in the address bar & press enter. Click on "***I will be Careful Promise***" **button**, in the search field type "search" which will display following preference ***"browsehelperApps.neverAsk.saveToDisk"***.

Each setting of the browser is called as "Preference" & set of these preferences (group of preference) are called as "**Profile**".

For the above preference if we specify ***"application/zip"*** it will download the zip file automatically without displaying File Download Popup. The above value is called as **MIME** type **(Multipurpose Internet Mail Extension)** we can get the **MIME** type from following website.

http://www.hansenb.pdx.edu/DHKB/dict/tutorials/mimetyp.php

```
Ex: Downloading the zip file into D drive
Note: By default the file will be saved in "Downlaods" folder.
```

Code:

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxProfile;

public class FileDownload
{
    public static void main(String[] args)
    {

    FirefoxProfile profile=new FirefoxProfile();
    //Dont ask the user if the file is .zip save it to disk
    profile.setPreference
    ("browser.helperApps.neverAsk.saveTodisk", "application/zip");
    //where in the disk, 0->Desktop; 1->Downloads; 2->Others
    profile.setPreference("browser.download.folderList",2);
    //if 2 then which location?
    profile.setPreference("browser.download.dir", "d:\\");
    //open Firefox with the above profile
    WebDriver driver=new FirefoxDriver(profile);
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    //Go to selenium download page
    driver.get("http://www.selenium.org/download/");
    //click on "Download" link of Java
    driver.findElement(By.xpath("//tr[td[text()='Java']]/td[4]/a")).click();
    }
}
```

Child Browser Popup:

Characteristics:

There will be a word "***WebDriver***" in the right bottom corner of the pop up.

Solution:

In order to handle the ***child browser popup***, we should transfer the control from parent browser into child browser using following statement.

```
driver.switchTo().window(String);
```

In the above statement "**String**" refers to "**WindowHandle**" of the browser.

"**WindowHandle"** is a unique alphanumeric string generated by the OS for every browser window.

In order to get the "WindowHandle" we can use following statement but it returns "WindowHandle" of current browser (by default parent window).

```
String a = driver.getWindowHandle();
```

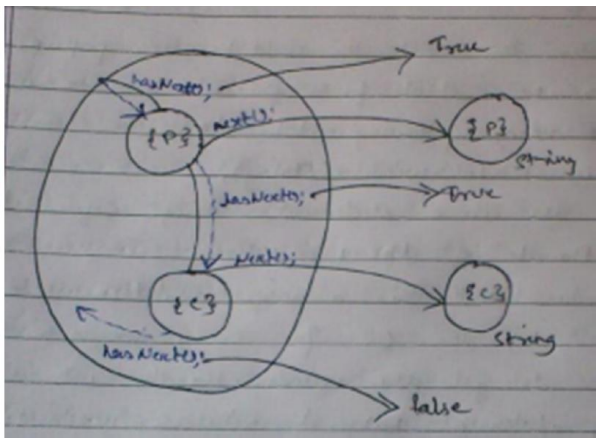
In selenium there is no option to directly get the "WindowHandle" of child browser, we can use following statement which returns "WindowHandle" of all the browsers (Parent & all the child browser).

```
Set<string> a = driver.getWindowHandles();
```

In order to get the string present in the set we use ***next()*** method of iterator.

Code:

```
package com.qspiders;

import java.util.Iterator;
import java.util.Set;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class ChildBrowserPopup
{
    public static void main(String[] args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("http://www.naukri.com/");
    Set<String> a = driver.getWindowHandles();
    Iterator<String> b=a.iterator();
    System.out.println(b.hasNext());
    System.out.println(b.next());
    System.out.println(b.hasNext());
    System.out.println(b.next());
    System.out.println(b.hasNext());
    }
}
```



Q. Write a code to count the number of browsers which are opened by the webdriver.

Code:

```
package com.qspiders;

import java.util.Set;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class CountBrowserWindow
{
    public static void main(String[] args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("http://www.naukri.com/");
    Set<String> a=driver.getWindowHandles();
    System.out.println("Number of Browsers: "+a.size());
    }
}
```

Q. Write a code to print title of all the browsers.

Code:

```
package com.qspiders;

import java.util.Iterator;
import java.util.Set;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class TitleofBrowser
{
    public static void main(String[] args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("http://www.naukri.com");
    Set<String> a =driver.getWindowHandles();
    Iterator<String> b=a.iterator();
    while(b.hasNext())
        {
        String w=b.next();
        driver.switchTo().window(w);
        System.out.println(driver.getTitle());
        }
    }
}
```

Output:

Naukri.com-Jobs in India….

UTC Aerospace

Q. Write a code to close only the parent browser.

Code:

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class CloseParentBrowser
{
    public static void main(String[] args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("http://www.naukri.com");
    driver.close();
    }
}
```

Q. Write a code to close all the browsers.

Code:

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class CloseAllBrowser
{
    public static void main(String[] args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("http://www.naukri.com");
    driver.quit();
    }
}
```

Q. Write a code to close only the child browser.

Code:

```
package com.qspiders;

import java.util.Iterator;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class CloseChildBrowser
{
    public static void main(String[] args)
    {
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driver.get("http://www.naukri.com");
        Iterator<String> b=driver.getWindowHandles().iterator();
        String parent=b.next();
        String child=b.next();
        driver.switchTo().window(child);
        driver.close();
        //Note: After closing the child browser we should switch
        //back to parent window or else we get
        //NoSuchWindowException
        driver.switchTo().window(parent);
        driver.close();
    }
}
```

Assignment:

Q. Write a code to print the "WindowHelp" of all the browser.

Q. Write a code to close all the browsers without using "*quit() method*".

Window Popup:

If the popup displayed in the application doesn't belongs to any of previous categories such as 'Alert & Confirmation', 'Hidden Division', 'Page on Load', 'File upload', 'File Download' & 'Child Browser Pop ups', then it is considered as "Window Popup". In selenium there is no option to handle the window popup we use third party tools such as "**AutoIT**".

Q. How do you run or execute an exe file in selenium?

```
try
{
Runtime.getRuntime().exec("calc");
}
catch(IOException e)
{
e.printStackTrace();
}
```

Installing AutoIT:

1. Open the browser & go to following website:

http://www.autoitscript.com/site/autoit/downloads/

2. Click on Download AutoIT then click save file, which will save the following file in Downloads folder.

"autoit-u3-setup.exe".

3. Double click on above file, click "Yes", click "Next", Click "I Agree", click "Next", click "Install".

4. Go to "***Start***" & go to "***All Programs***", expand "***AutoIT***" & select **"SCITE Script Editor"**.

This window is used to write the AutoIT script.

Write the following code & save it in the required location.

```
WinWaitActive("Opening selenium")
Send("LEFT")
Sleep(1000)
Send("{ENTER}")
```

Note:

The above script file will be saved with .au3 file extension. The “WinWaitActive” is used to activate the popup window, it automatically wait till the popup is displayed.

Go to “Tools” & select the option “Compile” which will create an exe file in the location where .au3 is present.

```java
package com.qspiders;

import java.io.IOException;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class AutoIt1
{
    public static void main(String[] args) throws IOException
    {
    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
    driver.get("http://www.seleniumhq.org/download/");
    driver.findElement(By.linkText("2.39.0")).click();
    String path="d:\\WindowPopup.exe";
    Runtime.getRuntime().exec(path);

    }
}
```

Page On load Popup:

Characteristics:

1. The popup is displayed while loading the webpage.
2. It is same as Confirmation popup & it will have username & password fields.

Solution:

In selenium we don't have any option to handle **"page on load popup"** we use **"AutoIT"** tool.

Open "***AutoIT Script Editor***", write the following script & save it.

```
WinWaitActive("Authentication Required")
Sleep(1000)
Send("admin")
Sleep(1000)
Send("{TAB}")
Sleep(1000)
Send("Admin")
Send("admin123")
Sleep(1000)
Semd("{ENTER}")
```

Write the following code in the main method.

```
package com.qspiders;

import java.io.IOException;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class AutoIT2
{
    public static void main(String[] args) throws IOException
    {
        Runtime.getRuntime().exec("d:\\POP.exe");
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
        driver.get("//192.168.1.1");
    }
}
```

Handling Frames:

In order to create embedded webpage, Developer uses <iframe> html tag, by default the webdriver searches for the element in the main page only. In order to perform any action on the element we should transfer the control from main page into the frame which can be done using any of the following statement.

```
driver.switchTo().frame(int);           'index starts from 0'
driver.switchTo().frame(String);        'name of the frame'
driver.switchTo().frame(WebElement);    'frame Element'
```

After performing the action in order to perform the action on any element which is present on the main page we should transfer the control back using the following statement.

```
driver.switchTo().defaultContent();
```

Ex:

1. Open the notepad & write the following code & save it as page1.html on the Desktop.

```
<html>
 <body>
   <iframe id="f1" src="page2.html"></frame></br>
   A1:<input type="text" id="a1">
  </body>
 </html>
```

2. Open the notepad & write the following code & save it as page2.html on the Desktop.

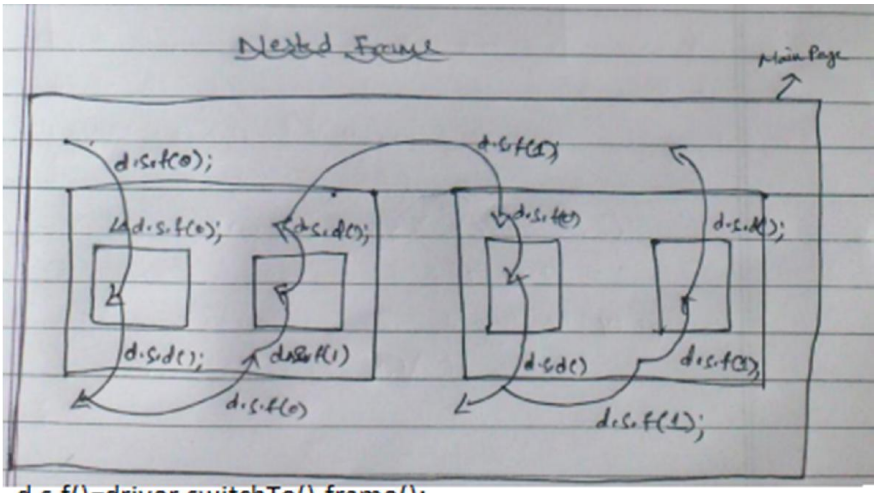
```
<html>
 <body>
      A2:<input type="text" id="a2">
  </body>
 </html>
```

3. Write the following code:

```java
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class AutoIT3
{

    public static void main(String[] args)
    {
    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
    driver.get("file:///C:/users/Waseem/Desktop/page1.html");
    driver.switchTo().frame(0);
    driver.findElement(By.id("a2")).sendKeys("xyz");
    driver.switchTo().defaultContent();
    driver.findElement(By.id("a1")).sendKeys("abcd");
    }
}
```

Note:

If the element is inside the frame then the browser will display “This Frame” in Context menu when we right click on that element.



d.s.f()=driver.switchTo().frame();
d.s.d()=driver.switchTo().Defaultcontent();

Ex: Uploading CV in naukri.com

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class NestedFrame1
{
    public static void main(String[] args)
    {
    WebDriver driver = new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("http://www.naukri.com/");
    driver.findElement(By.linkText("Post Resume")).click();
    driver.switchTo().frame("uploadframe");
    driver.findElement(By.id("browsecv")).sendKeys("d:\\Mycv.docx");
    }
}
```

Handling Excel File:

While testing the application we need to verify the feature with valid & invalid values. We always take these inputs from external source such as text file, CSV file, Database file & XML.

The very frequently used file used is Excel file. In order to read data from excel file & write data to excel file we can use many of the java API's very frequently used one is **"Apache POI" (Poor Obfuscation Implementation).**

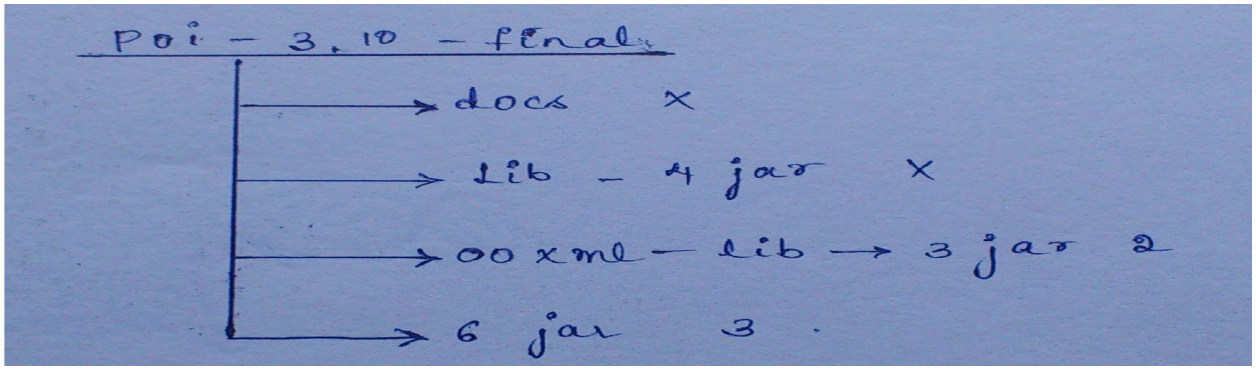
It can be downloaded from following website:

http://poi.apache.org/download.html(#POI-3.10-FINAL

Under Binary Distribution we have to download "poi-bin-3.10-FINAL-20140208.zip"

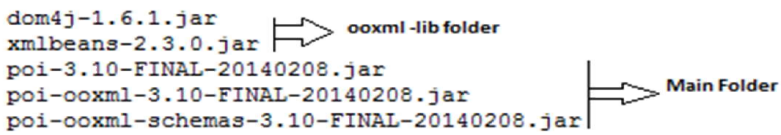
After downloading the above file, copy & paste it to required location & unzip it or extract it, which will create a folder "**poi-3.10-FINAL**".

In the above folder there will be totally 13 jar files out of which only 5 are mandatory.



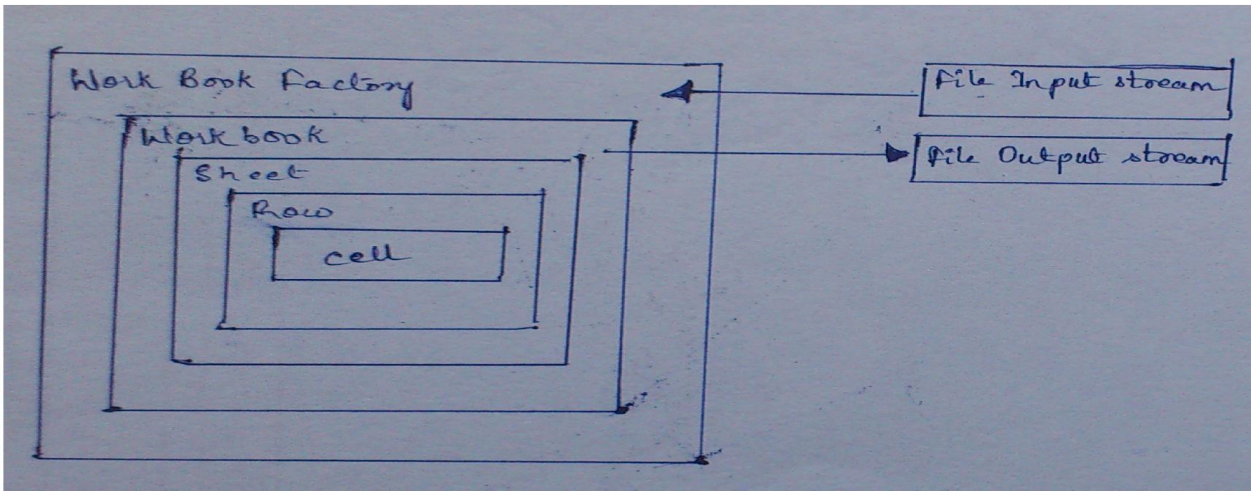
Create a folder with the name “POI” in the required location.

Copy & paste following files,



Open the ***Eclipse IDE***, right click on the ***“java project”***, go to ***“Build Path”*** & select ***“Add External Archives”***, navigate to the newly created ***POI folder*** & select all the 5 jar files & click open.

Simplified Architecture of Apache POI:



Note:

Ensure that OS is displaying the extension of the files.

Ex: Reading the value present in first row, first cell of Sheet1.

```
package com.qspiders;

import java.io.FileInputStream;
import java.io.IOException;

import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;

public class ExcelFile1
{
    public static void main(String[] args) throws InvalidFormatException, IOException
    {
    FileInputStream fis=new FileInputStream("F:/Book1.xlsx");
    Workbook wb = WorkbookFactory.create(fis);
    Sheet s = wb.getSheet("Sheet1");
    Row r = s.getRow(0);
    Cell c = r.getCell(0);
    String v = c.getStringCellValue();
    System.out.println(v);
    }
}
```

```
Optimised Code: String=s.getRow(0).getCell(0).getStringCellValue();
```

Note: If Sheet Name is invalid or row number or cell number are invalid in such cases we get "**NullPointerException**".

Q. Write a code to print all the cell values of first row of an excel sheet.

```
package com.qspiders;

import java.io.FileInputStream;
import java.io.IOException;

import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;

public class ExcelFile2
{
    public static void main(String[] args) throws InvalidFormatException, IOException
    {
    //.....................................
    FileInputStream fis=new FileInputStream("F:/Book1.xlsx");
    Workbook wb = WorkbookFactory.create(fis);
    Sheet s = wb.getSheet("Sheet1");
    Row r = s.getRow(0);
    int cc = r.getLastCellNum();
    System.out.println(cc);
    for(int i=0; i<cc; i++)
        {
        String v=r.getCell(i).getStringCellValue();
        System.out.println(v);
        }
    }
}
```

Q. Write a code to print the number of rows & number of cells in each row & also print the content of the excel sheet.

```
package com.qspiders;

import java.io.FileInputStream;
import java.io.IOException;
import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;

public class ExcelFile3
{
    public static void main(String[] args) throws InvalidFormatException, IOException
    {
    FileInputStream fis=new FileInputStream("F:/Book1.xlsx");
    Workbook wb = WorkbookFactory.create(fis);
    Sheet s = wb.getSheet("Sheet1");
    int rc = s.getLastRowNum(); //returns index of Last Row
    System.out.println("Row Count: "+rc);
    for(int i=0; i<=rc; i++)
        {
        Row r = s.getRow(i);
        int cc = r.getLastCellNum(); //returns count
        System.out.println("Row:"+i+ " Cell Count: "+cc);
        for(int j=0; j<cc; j++)
        {
            String v = r.getCell(j).getStringCellValue();
            System.out.print(v+ " ");
        }
        System.out.println();
        }
    }
}
```

```
package com.qspiders;

import java.io.FileInputStream;
import java.io.IOException;
import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;

public class ExcelFile3
{
    public static void main(String[] args) throws InvalidFormatException, IOException
    {
    FileInputStream fis=new FileInputStream("F:/Book1.xlsx");
    Workbook wb = WorkbookFactory.create(fis);
    Sheet s = wb.getSheet("Sheet1");
    int rc = s.getLastRowNum(); //returns index of Last Row
    System.out.println("Row Count: "+rc);
    for(int i=0; i<rc; i++)
        {
        Row r = s.getRow(i);
        int cc = r.getLastCellNum(); //returns count
        System.out.println("Row:"+i+ " Cell Count: "+cc);
        for(int j=0; j<cc; j++)
        {
        String v = r.getCell(j).getStringCellValue();
        System.out.println(v+ " ");
        }
        System.out.println();
        }
    }
}
```

Writing data into Excel Sheet:

While writing data into excel sheet we use ***write () method*** of ***workbook*** which takes an argument of type "***FileOutputStream"***.

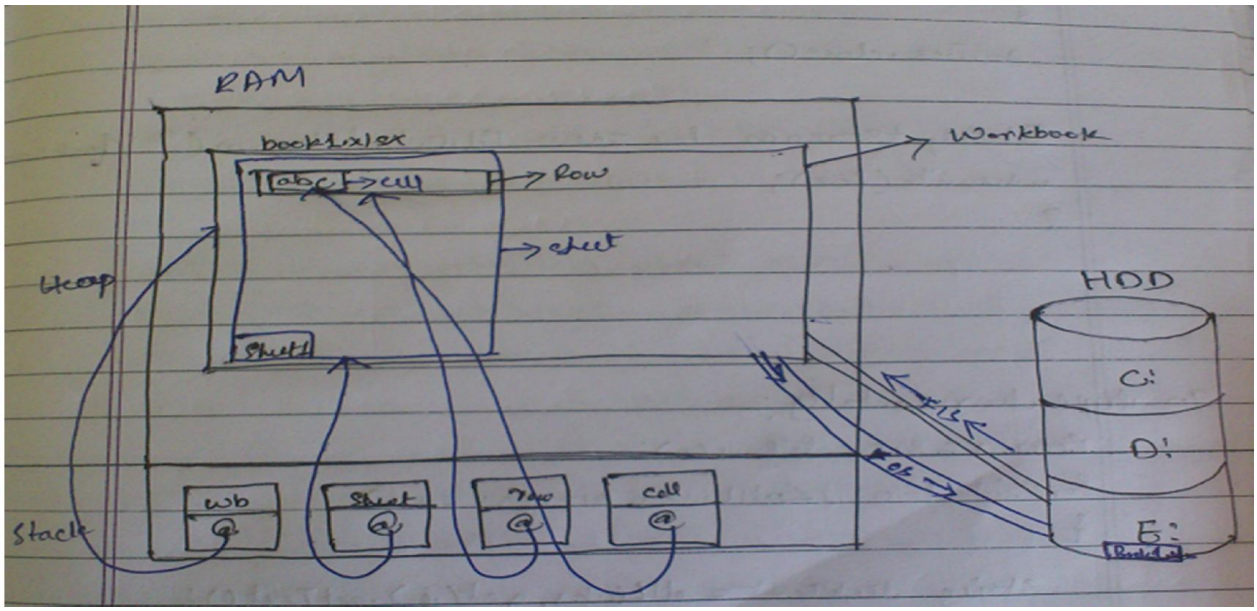
Before writing the data into Excel file we should open it using "***FileInputStream***" as shown below.

```
package com.qspiders;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;

public class WrExcel4
{
    public static void main(String[] args) throws InvalidFormatException, IOException
    {
    FileInputStream fis=new FileInputStream("F:/Book1.xlsx");
    Workbook wb = WorkbookFactory.create(fis);
    Sheet s = wb.getSheet("Sheet1");
    Row r = s.getRow(0);
    Cell c = r.getCell(0);
    c.setCellValue("Java");
    FileOutputStream fos=new FileOutputStream("F:/Book1.xlsx");
    wb.write(fos);
    }
}
```

Note:

- The above code is used to write the data into an existing cell.
- In order to write the data into a new cell then we should use "***create cell method***".
- In order to write the data into new row then we should use "***create row method***" as show below.

```
package com.qspiders;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;

public class ExcelFile5
{
    public static void main(String[] args) throws InvalidFormatException, IOException
    {
    FileInputStream fis=new FileInputStream("F:/Book1.xlsx");
    Workbook wb = WorkbookFactory.create(fis);
    Sheet s = wb.getSheet("Sheet1");
    s.getRow(0).createCell(0).setCellValue("waseem");
    FileOutputStream fos=new FileOutputStream("F:/Book1.xlsx");
    wb.write(fos);
    }
}
```



Q. Write a code to get name of all the links present on the webpage & write it into Excel Sheet.

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

public class ExcelFile6
{
    public static void main(String[] args) throws InvalidFormatException, IOException
    {
    FileInputStream fis=new FileInputStream("F:/Book1.xlsx");
    Workbook wb = WorkbookFactory.create(fis);
    Sheet s = wb.getSheet("Sheet1");

    WebDriver driver=new FirefoxDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("file:///C:/Users/Waseem/Desktop/mypage.html");
    List<WebElement> allLinks = driver.findElements(By.xpath("//a"));
    for(int i=0; i<allLinks.size(); i++)
    {
        String linkText = allLinks.get(i).getText();
        s.createRow(i).createCell(0).setCellValue(linkText);
    }
    driver.close();
    FileOutputStream fos=new FileOutputStream("F:/Book1.xlsx");
    wb.write(fos);
    }
}
```

Q. To type horizontally:

```
public class ExcelFile7
{
    public static void main(String[] args) throws InvalidFormatException, IOException
    {
        FileInputStream fis=new FileInputStream("F:/Book1.xlsx");
        Workbook wb = WorkbookFactory.create(fis);
        Sheet s = wb.getSheet("Sheet1");

        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driver.get("file:///C:/Users/Waseem/Desktop/mypage.html");
        List<WebElement> allLinks = driver.findElements(By.xpath("//a"));

        for(int i=0; i<allLinks.size(); i++)
        {
            String linkText = allLinks.get(i).getText();
            r.createRow(0).createCell(i).setCellValue(linkText);
        }
        driver.close();
        FileOutputStream fos=new FileOutputStream("F:/Book1.xlsx");
        wb.write(fos);
    }
}
```

Something wrong in the code:

```
Row r=s.createRow(0);
for(int i=0; i<allLinks.size(); i++)
{
String linkText=allLinks.get(i).getText();
r.createRow(0).createCell(i).setCellValue(linkText);
}
```

Q. How do you test the login feature by taking test data from excel sheet.

Step 1: Create an excel file in the required location.

Ex: Create “Book1.xlsx” in F: Drive.

Enter the values in Sheet 1 as show below.

UserName	Password	eTitle	aTitle	Status
admin	admin	actitime-EnterTimeTrack		
user	user	actitime-EnterTimeTrack		
guest	guest	actitime-EnterTimeTrack		

```
    {
    FileInputStream fis = new FileInputStream("F:/Book1.xlsx");
    Workbook wb = WorkbookFactory.create(fis);
    Sheet s = wb.getSheet("Sheet1");
    int rc = s.getLastRowNum();
    String un,pw,eTitle,aTitle,result;
    for(int i=1; i<rc; i++)
    {
        un=s.getRow(i).getCell(0).getStringCellValue();
        pw=s.getRow(i).getCell(1).getStringCellValue();
        eTitle=s.getRow(i).getCell(2).getStringCellValue();
        WebDriver driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driver.get("http://demo.actitime.com");
        driver.findElement(By.name("username")).sendKeys(un);
        driver.findElement(By.name("pwd")).sendKeys(pw);
        driver.findElement(By.id("loginButton")).click();
        Thread.sleep(5000); //implicitlyWait will not work for getTitle
        aTitle=driver.getTitle();
        if(eTitle.equals(aTitle))
        {
            result="Pass";
        }
        else
        {
            result="Fail";
        }
        driver.close();
        s.getRow(i).createCell(3).setCellValue(aTitle);
        s.getRow(i).createCell(4).setCellValue(result);
    }//ends of for loop
    FileOutputStream fos=new FileOutputStream("F:/Book1.xlsx");
    wb.write(fos);
    }
}
```

Working with Internet Explorer Browser:

1. Go to following website:
   http://code.google.com/p/selenium/downloads/list
2. Based on the operating system type download the following file:
   ```
   IEDriverServer_x64_2.39.0.zip   - for 64Bit OS
   IEDriverServer_win32_2.39.0.zip - for 32Bit OS
   ```
3. After downloading the required zip file copy & paste it to required location & extract it, which will create following "exe" file.
   ```
   IEDriverServer.exe
   ```
4. Open IE browser ensure that zoom level is 100% (Ctrl+0).
5. Go to ***Tools*** & select ***Internet Options***, Go to ***Security*** Tab & select "***Enable Protected Mode***" checkbox in all the following zones.
   - **i. Internet**
   - **ii. Local Intranet**
   - **iii. Trusted Sites**
   - **iv. Restricted Sites**

   And then click "**Apply**" & "**OK**".

Write the following code & execute.

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.ie.InternetExplorerDriver;

public class IEBrowser
{
    public static void main(String[] args)
    {
    System.setProperty("webdriver.ie.driver", "F:/IEDriverServer.exe");
    WebDriver driver=new InternetExplorerDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("http://demo.actitime.com");
    driver.findElement(By.name("username")).sendKeys("admin");
    driver.findElement(By.name("pwd")).sendKeys("manager");
    driver.findElement(By.id("loginButton")).click();
    }
}
```

Note:

1. While executing the script on IE browser, the IEDriverServer.exe file will print the log information in console & it will be in red color.
2. Other than Mozilla Firefox the word WebDriver will not be displayed on the browser.
3. While writing the script using IE browser in order to inspect the element we should use IE Developer Tool (F12).

Working with Chrome Browser:

1. Go to the following website:
   http://chromedriver.storage.googleapis.com/index.html
2. Click on the folder which has latest version (2.9).
3. Download the following file:
   `chromedriver_win32.zip`
4. Copy & paste the zip file to the required location & extract it, which will create following file.
   `chromedriver.exe`

Write the following code:

```
package com.qspiders;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class ChromeBrowser
{
    public static void main(String[] args)
    {
    System.setProperty("webdriver.chrome.driver","F:/chromedriver.exe");
    WebDriver driver=new ChromeDriver();
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    driver.get("http://demo.actitime.com");
    driver.findElement(By.name("username")).sendKeys("admin");
    driver.findElement(By.name("pwd")).sendKeys("manager");
    driver.findElement(By.id("loginButton")).click();
    }
}
```

TestNG: "*Test Next Generation*" is a Unit Testing tool basically used by developers to perform White Box Testing.

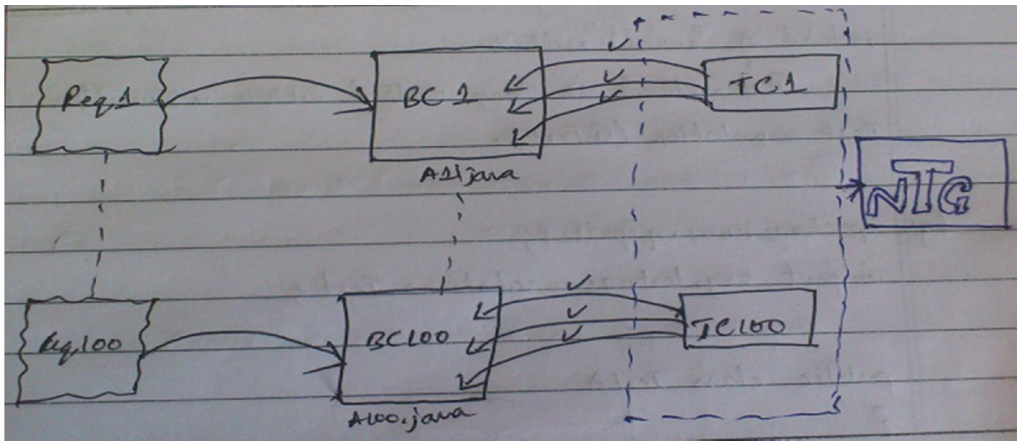
We can also use the same tool to perform Black Box Testing.

Generally for every requirement Developer develops a java class which is called as "***Business Class***".

Before giving the build to the testing team developer should perform Unit Testing to ensure that Business class is working properly.

Testing every business class manually is monotonous & time consuming; hence Developer develops another java class to check the business class which is called as "***Test Class***".

The Test Classes are developed with the help of Unit Testing tool such as "**testNG**", so that we can execute all the test classes together. We see the execution result & also can re-run only failed test classes after debugging it.



In selenium for every manual test case, we write Automation script i.e. a java class, so in order to execute all the java classes in order to get the execution result etc. we will also use TestNG.

We can install TestNG as plug in for Eclipse IDE.

Installing TestNG:

1. Open the ***Eclipse IDE*** & go to "***Help***" & select "***Eclipse Market Place***."
2. Type "***TestNG***" in ***Find field*** press enter which will display "***TestNG***" for eclipse, click on the ***Install*** button.
   Click "***Next"***, select "***I accept***" radio button, click "***Finish***".
3. Click "***OK***" on warning message. Click "***Yes***" on the confirmation popup which will close & reopens the Eclipse IDE.
4. Right click on the ***java project***, Go to Build path, select "***Add Libraries***". Select "***TestNG***", click "***Next***" & click "***Finish***", which will associate "***testNG.jar***" with the ***java project***.

TestNG Class:

"TestNG" class is a java class which contains "test method" instead of "main()" method.

Test method is any method which is written below "***Test annotation (@Test)***"

```
package com.qspiders;

import org.testng.annotations.Test;

public class TestA
{
@Test
public void testA()
    {
        System.out.println("Login");
        System.out.println("Logout");
    }
}
```

In order to execute the TestNG class, we can click on the ***Run button*** present on the ***Eclipse IDE*** or right click on the ***Java class***, Go to ***Run As*** & select ***TestNG Test*** after the execution it will print the result in console as show below.

Output:

```
Login
Logout
PASSED: testA

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================
```

After executing TestNG class the TestNG will automatically generate the result in **HTML format.**

In order to see it, right click on the ***Java Project***, select ***Refresh*** which will display "***test-output***" folder. Expand; right click "***index.html***" or "***emailable-report.html***". Go to ***open with*** & select **"WebBrowser"**.

Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
Default suite						
Default test	1	0	0	46		

Q. How do you write the information into TestNG html report & as well as into the console.

A: ***Using log method of Reporter class.***

Q. Can we have multiple test methods in TestNG class?

A: Yes. In a TestNG class we can have multiple test methods & by default it will execute the test method in alphabetical order.

Q. Can we execute a test before executing other test if it is not in alphabetical order?

A: Yes it is possible using "**dependsOnMethods**" parameter of ***test annotation(@test).***

```
package com.qspiders;

import org.testng.Reporter;
import org.testng.annotations.Test;

public class TestA1 {

@Test
    public void testB()
    {
        Reporter.log("Create customer",true);
    }
@Test(dependsOnMethods={"testB"})
    public void testA()
    {
        Reporter.log("Delete customer",true);
    }
}
```

Q. What happens to the test method when the other test method on which it is depended fails.

A: TestNG will skip the test method.

Ex: If testA() method depends on testB() method & if testB() method fails then TestNG will skip testA() method.

Q. What is "annotation" & what are the important annotations which are used in selenium.

A: Annotation is a syntactic metadata. Annotations are the predefined words which starts with "@" symbol.

In TestNG annotation basically indicates the type of the method & their execution order.

Following 3 are very important annotations used in TestNG while developing selenium script.

```
@BeforeMethod
@AfterMethod
@Test
```

Note: "Before Method" will be executed before every test method & "After Method" will be executed after every test method.

Ex:

```
package com.qspiders;

import org.testng.Reporter;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

public class TestABC
{
@BeforeMethod
public void preCondition()
    {
        Reporter.log("login",true);
    }
@AfterMethod
public void postCondition()
    {
        Reporter.log("logout",true);
    }

@Test
public void testA()
    {
        Reporter.log("Create customer",true);
    }
@Test
public void testB()
    {
        Reporter.log("Delete customer",true);
    }
}
```

Output:

```
login
Create customer
logout
login
Delete customer
logout
PASSED: testA
PASSED: testB

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================
```

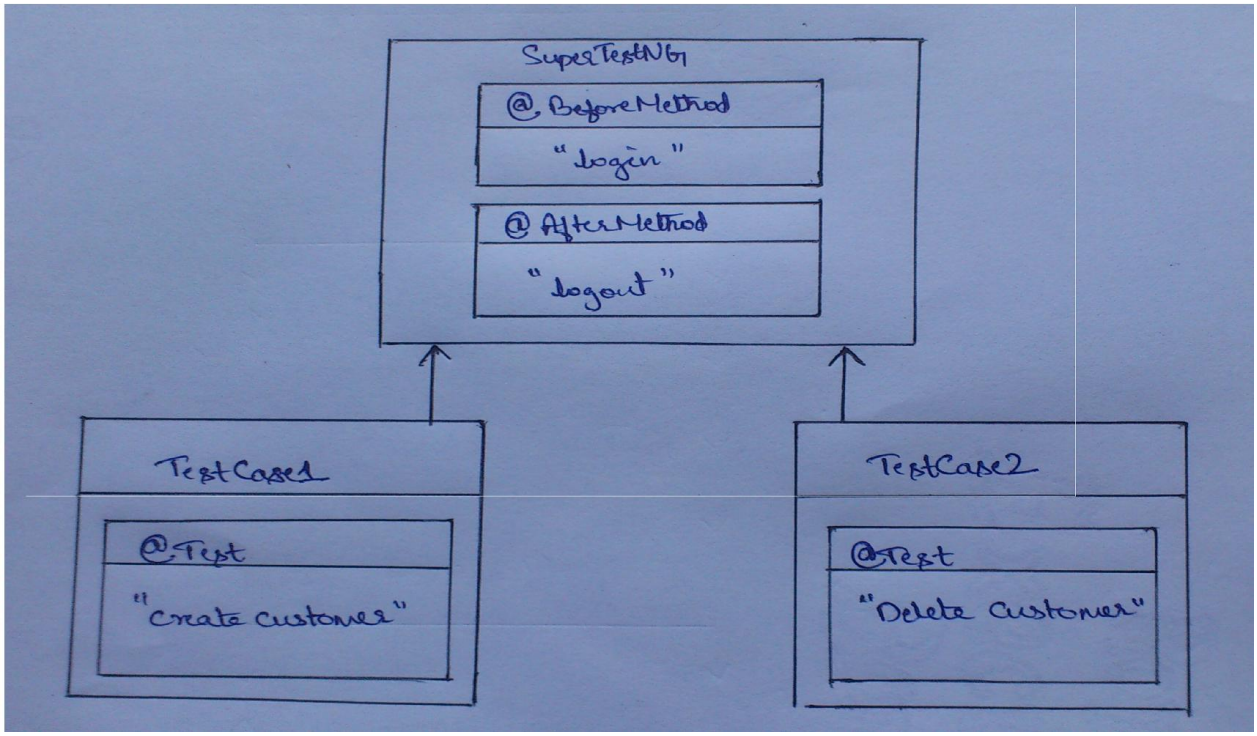
Inheritance in TestNG:

While developing selenium script we create a TestNG class for every manual test case.

In this scenario we must develop "***BeforeMethod***" & "***AfterMethod***" in every TestNG class, which will not only consumes space but also increases the time & maintenance.

In order to overcome this we use ***Inheritance*** which will be done by another "***extends***" keyword. The **first class** is called as "***Super class" or "Parent class" or "Base class***" whereas the **second class** is called as "***Sub class" or "child class"***.

In TestNG we create parent class where we develop the "***BeforeMethod***" & "***AfterMethod***" and in all the TestNG class we extend this parent class.



```java
package com.qspiders;
import org.testng.Reporter;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
//--------------------SuperTestNG.java----------------------
public class SuperTestNG
{
    @BeforeMethod
    public void preCondition()
    {
        Reporter.log("Login",true);
    }
    @AfterMethod
    public void postCondition()
    {
        Reporter.log("Logout",true);
    }
    //--------------------TestCase1.java----------------------
    public class TestCase1 extends SuperTestNG
    {
        @Test
        public void testA()
        {
            Reporter.log("Create customer",true);
        }
    }
    //--------------------TestCase2.java----------------------
    public class TestCase2 extends SuperTestNG
    {
        @Test
        public void testB()
        {
            Reporter.log("Delete customer",true);
        }
    }
```

Output:

```
Login
Create customer
Logout
Login
Delete customer
Logout
PASSED: testA
PASSED: testB

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================
```

TestNG-Suite:

In order to execute multiple TestNG classes we need to create ***TestNG suite***.

TestNG suite is an ***XML (Extensible Markup Language)*** file which contains list of all the TestNG class which are to be executed.

Ex:

```
<suite name="Suite" parallel="none">
  <test name="Test">
    <classes>
      <class name="com.qspiders.TestA"/>
      <class name="com.qspiders.TestCase2"/>
      <class name="com.qspiders.SuperTestNG"/>
      <class name="com.qspiders.TestCase1"/>
      <class name="com.qspiders.TestABC"/>
      <class name="com.qspiders.TestA1"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

In order to create TestNG file, right click on ***Java Project***, go to ***TestNG***, ***select Convert to TestNG*** & click "***Finish***". It will create ***TestNG.xml file*** inside ***the Java Project folder***.

Q. What are the other annotations supported by TestNG?

```
A: @DataProvider
   @BeforeClass
   @AfterClass
   @BeforeTest
   @AfterTest
   @BeforeSuite
   @AfterSuite
```

Performing Regional Regression Testing using TestNG:

When we receive a build & execute all the TestNG classes it will be called as "***Full Regression Automation Testing***".

But in order to save the time there are many times we execute only the test cases which belongs to the impact area, this is called as "***Regional Regression Testing***".

In order to do this we use ***TestNG groups***.

In order to create TestNG Group we use ***groups parameter*** as shown below.

```java
public class RRTestCases
{
    @BeforeMethod(groups={"all"})
    public void preCondition()
    {
        Reporter.log("login",true);
    }
    @AfterMethod(groups={"all"})
    public void postCondition()
    {
        Reporter.log("logout",true);
    }
    @Test(groups={"customer","smoke"})
    public void testA()
    {
        Reporter.log("Create customer",true);
    }
    @Test(groups={"customer"})
    public void testB()
    {
        Reporter.log("Delete customer",true);
    }
    @Test(groups={"Product","smoke"})
    public void testC()
    {
        Reporter.log("Create Product",true);
    }
    @Test(groups={"Product"})
    public void testD()
    {
        Reporter.log("Delete Product",true);
    }
}
```

In the above TestNG class we have following 4 groups.

```
1. all -> BeforeMethod & AfterMethod
2. customer -> TestA & TestB
3. Product -> TestC & TestD
4. Smoke -> TestA & TestC
```

In order to execute the TestNG class or test method which belongs to required group we use <include> tag as show below in TestNG suite.

```xml
<suite name="Suite" parallel="none">
  <test name="Test">
  <groups>
   <run>
    <include name="all"/>
    <include name="customer"/>
   </run>
  </groups>
   <classes>
     <class name="com.qspiders.RRTestCases"/>
    </classes>
  </test>
</suite>
```

Execution Result:

Tag:

```
<include name="all"/>
<include name="customer"/>
```

Output:

```
login
Create customer
logout
login
Delete customer
logout
```

Tag:

```
<include name="all"/>
<include name="Product"/>
```

Output:

```
login
Create Product
logout
login
Delete Product
logout
```

Tag:

```
<include name="all"/>
<include name="smoke"/>
```

Output:

```
login
Create customer
logout
login
Create Product
logout
```

Tag:

```
<exclude name="smoke"/>
```

Output:

```
login
Delete customer
logout
login
Delete Product
logout
```

Parallel Execution of Tests using TestNG:

In order to execute the TestNG classes parallelly we specify ***parallel="classes"*** in TestNG suite.

```
public class TestCase1
{
@Test
public void testA() throws InterruptedException
    {
    WebDriver driver=new FirefoxDriver();
    driver.get("http://demo.actitime.com/login.do");
    for(int i=1; i<10; i++)
        {
        driver.findElement(By.name("username")).sendKeys("admin"+i);
        Thread.sleep(1000);
        driver.findElement(By.name("username")).clear();
        }
    driver.quit();
    }
}
```

```
public class TestCase2
{
    @Test
    public void testB() throws InterruptedException
    {
        WebDriver driver=new FirefoxDriver();
        driver.get("http://127.0.0.1/login.do");
        for(int i=1; i<10; i++)
        {
            driver.findElement(By.name("username")).sendKeys("manager"+i);
            Thread.sleep(1000);
            driver.findElement(By.name("username")).clear();
        }
        driver.quit();
    }
}
```

XML Code:

```
<suite name="Suite" parallel="classes">
  <test name="Test">
    <classes>
      <class name="com.qspiders.TestCase1"/>
      <class name="com.qspiders.TestCase2"/>
    </classes>
  </test>
</suite>
```

Handling Database:

While testing the application if we cannot verify any information in user interface we may go & verify in databse.

Before verifying anything in the database we should have following information of database.

1. Data type
2. Database location
3. Username & password
4. Table Name & its structure (schema).

Ex: For ActiTime application:

1. Database type is MS Access.
2. Database location is "C:\Program Files\actiTime\database
3. User & password – For this database there is no username & password.
4. Table Name is "customer".
5. Table structure

```
FieldName
id
create_timestamp
name
name_lower
description
archiving_timestamp
```

While connecting to the database we need to specify the above information which is called as "connection string" instead of typing the connection sting every time we can store it in a system variable called ***DSN (Data Source Name).***

Steps to create DSN:

1. Go to ***start*** & type "***data***" select **"Data Sources (ODBC)".**
2. Go to "**System DSN**" tab, click on "***Add***", select the required database type (***Microsoft Access Driver*.mdb***) click **Finish**.
3. Specify the DSN name
   Ex: actiTime
4. Click **Select** & navigate to the location where database is present ***C:/Program Files/actiTime/database***
5. Select the ***database file name*** & click ***Ok***.
6. If the database is **secure** then click on ***Advanced*** where you can specify the **login name & password**. And click **OK**.

Note:

1. **ODBC** stands for ***Open Database Connectivity.***
2. In order to connect to specific database we should ensure that the required software is installed which is called as **Database driver**.
3. Creating DSN will be different for different database.

Accessing Database Programmatically:

In order to connect to database we use DSN as shown below.

```
package com.qspiders;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DemoA
{
    public static void main(String[] args) throws SQLException
    {
        //open connection to DB
        Connection c = DriverManager.getConnection("jdbc:odbc:actiTIME");
        //close the connection
        c.close();
        System.out.println("done");
    }
}
```

Q. Write a code to count the number of columns present in the specified table of database & also print name of all the columns present in the database.

```
package com.qspiders;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;

public class DB2
{
    public static void main(String[] args) throws SQLException
    {
    Connection c = DriverManager.getConnection("jdbc:odbc:actiTIME");
    String sql = "select * from customer";
    ResultSet r = c.createStatement().executeQuery(sql);

    int cc=r.getMetaData().getColumnCount();
    System.out.println("Columns: "+cc);
    for(int i=1; i<=cc; i++) //index starts from 1
        {
        String cn = r.getMetaData().getColumnName(i);
        System.out.println(cn);
        }
        c.close();
    }
}
```

Output:

```
Columns: 6
id
create_timestamp
name
name_lower
description
archiving_timestamp
```

Q. Write a code to print the content of specified table present in the database.

```
package com.qspiders;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;

public class DB3
{
    public static void main(String[] args) throws SQLException
    {
    Connection c = DriverManager.getConnection("jdbc:odbc:actiTIME");
    String sql="select * from customer";
    ResultSet r=c.createStatement().executeQuery(sql);

    int cc=r.getMetaData().getColumnCount();
    while(r.next())
    {
        for(int i=1; i<=cc; i++)
        {
            String v=r.getString(i);
                System.out.println(v);
        }
        System.out.println("------------------");
    }
    c.close();
    }
}
```

Output:

```
6
2014_02_05      08:44:09
ABC
abc

null
------------------
7
2014-02-05      08:51:23
Apple
apple

null
------------------
```

Automation Framework:

Automation Framework is a standard set of rules, guideline & best practices which are followed to efficiently convert manual testcases into Automation scripts.

Automation Framework has 3 major stages

1. Designing Automation Framework.

2. Implementing Automation Framework (Developing methods & automating testcases)

3. Execution of the Framework

Designing Automation Framework:

In this initial phase Lead & Senior Engineers will design the Automation Framework based on past experience in & domain knowledge.

In this phase we specify required software, folder structure & naming convention.

Ex: Required Software:

1. JDK
2. Eclipse IDE with TestNG plugin
3. Selenium Server Standalone jar file
4. Mozilla Firefox Browser with Firebug & Firepath Add-ons.
5. Google Chrome Browser with chromeDriver.exe file
6. IEDriver.exe file
7. Microsoft Office
8. Apache POI jar file
9. AutoIT

Ensure that all the above software's are latest versions.

Steps to create Automation Framework:

1. Go to required location, example ***D:\Driver*** & create a folder with the name "***Workspace***".

2. Open the Eclipse, go to ***File → Switch Workspace → Other Browser*** & select the above created Workspace folder & click ***OK*** which will close & reopens the Eclipse IDE.

3. Create the Java Project.
   Ex: ***File → New → Project*** & select ***Java Project***, click ***Next*** specify the Project Name as "***Automation***" & click "***Finish***" & click "***Yes***" on the confirmation message.
4. Associate **TestNG** library.
   Right click on ***Java Project***, go to ***Build Path***, select ***Add Libraries*** select ***TestNG*** click ***Next*** & click ***Finish***.

5. Right click on the ***Java Project***, Go to ***New***, select ***Folder*** specify the folder name as ***Jars*** & click ***Finish.***

6. Associate jar files that is Right click on ***Java Project,*** Go to ***Build Path,*** select ***Add External Archives,*** navigate to ***D:\Workspace\Automation\jars***, select all the jar files (Ctrl+A) & click ***open***.

7. Right click on the ***Java Project***, Go to ***New*** & select the folder, specify the folder name as "***exe files***" & click ***Finish***.
   Copy & paste "***chromedriver.exe***" & "***IEDriverserver.exe***" files into the above created folder.

Note:

In the above folder we should store any exe file which is required in the Automation.

Ex: Exe file of ***AutoIT*** is also saved in the above folder.

8. Right click on the ***Java Project***, Go to ***New***, select folder, specify the folder name as ***TestDatas*** & click ***Finish***. The above folder is used to store the data, most of the cases it will be ***Excel file***.

Folder Structure:

➔ **Automation**
→src
→JRE System Library
→ TestNG
→Referenced Libraries
→ExeFiles
→Jars
→TestDatas

Developing Methods:

Steps to Develop Method:

1. For any given requirement first write the code in main method.

2. While writing the code do not throw any ***exception*** always ***try catch*** it.

3. While writing the code do not use the values directly instead of that store the values in a variable & use those variables in the code.

4. Ensure that the proper code is been written in catch block also.

```java
package com.qspiders;

import java.io.FileInputStream;

import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;

public class DemoA
{
    public static void main(String[] args)
    {
    String xlPath="F:/Book1.xlsx";
    String sheetName="Sheet1";
    int rc;
    try{
        FileInputStream fis=new FileInputStream(xlPath);
        Workbook wb = WorkbookFactory.create(fis);
        Sheet s = wb.getSheet(sheetName);
        rc=s.getLastRowNum();
    }
    catch(Exception e)
    {
        rc=-1;
    }
    System.out.println(rc);
    }//end of main
}
```

5. After writing the code in the main method, execute it & ensure that it is working fine.

6. Create an empty method with user friendly name, initially the method argument will be blank & return type will be void.

```java
public void getExcelRowCount()
{

}
```

7. Make the variable that are initialized in the main method as argument for newly created method, then delete those variables from the main method.

8. Copy & paste the remaining code of main method into the newly created method.

9. In order to return any value from the method write the appropriate return statement & change the return type of the method from void to respective data type.

Ex:

```
    public int getExcelRowCount(String xlPath, String sheetName)
    {
    int rc;
    try{
        FileInputStream fis=new FileInputStream(xlPath);
        Workbook wb = WorkbookFactory.create(fis);
        Sheet s = wb.getSheet(sheetName);
        s.getLastRowNum();
        }
    catch(Exception e)
    {
        rc=-1;
    }
    return rc;
}//.....................
```

Q. Develop a method to return the value present in specified cell of the Excel sheet.

```
public String getExcelCellValue(String xlPath, String sheetName, int rowNum, int CellNum)
{
    String v;
try
{
    FileInputStream fis=new FileInputStream(xlPath);
     Workbook wb = WorkbookFactory.create(fis);
     Sheet s = wb.getSheet(sheetName);
     v=s.getRow(rowNum).getCell(CellNum).getStringCellValue();
}
catch(Exception e)
{
    v=""; //empty string --> string with length zero
}
return v;
}
```

Q. Write a method to wait for specified amount of seconds.

```
public class Demo
{
    public void explicitWait(int duriation)
    {
        try
        {
            Thread.sleep(duriation*1000);
        }
        catch(Exception e)
        {

        }
    }
}
```

Q. Write a method to execute the specified SQL statement & return the result set.

```
public ResultSet getResultSet(String dsn, String sql)
{
    ResultSet rs;
    try
    {
        Connection c=DriverManager.getConnection(dsn);
        rs=c.createStatement().executeQuery(sql);
    }
    catch(SQLException e)
    {
        rs=null;
    }
    return rs;
}
```

Q. Write a method to capture the screenshot of the application.

```
public void getScreenshot(WebDriver driver, String imgPath)
{
    EventFiringWebDriver eDriver=new EventFiringWebDriver(driver);
    File srcFile=eDriver.getScreenshotAs(OutputType.FILE);
    try
    {
        FileUtils.copyFile(srcFile, newFile(imgPath));
    }
    catch(IOException e)
    {

    }
}
```

Q. What are Generic & Project specific methods?

A: If the given method can be used in other Automation Project also then it is called as "***Generic Methods",*** or else it is called as ***"Project Specific Methods".***

Creating Generic Library:

We store generic methods in a separate class file, in order to do this

1. Right click on "***src***" folder, go to ***"New"*** & select ***"package"*** specify the package name as ***"com.projectname.libraries"*** & click "Finish".

2. Right click on above created package, Go to "***New***" select "***class***", specify the class name as ***Generic*** & click ***Finish*** (No main method).

3. Write all the generic methods inside the above class.

```
package com.projectname.libraries;

public class Generic
{
        public static void main(String[] args)

        public int getExcelRowCount(String xlPath, String sheetName)

        public String getExcelCellValue(String xlPath, String sheet)

        public ResultSet getResult(String dsn, String sql)

        public void explicitWait(int duriation)

        public void getScreenshot(WebDriver driver, String imgPath)

}//End of class
```

Developing Project Specific Libraries:

We write the methods for repeating steps which are present in the test cases. If it is specific to this project then we write the method inside ***Project Specific class*** or else it will be written inside ***Generic class***.

Sample Test Cases:

Test Case 1:
AT_Customer_Create
Steps:

i. Login to ***ActiTime*** application with valid username & password.
ii. Click on ***Tasks*** & then click on ***Project & Customers***.
iii. Click on ***Create Customer***.
iv. Enter valid ***Customer Name*** such as ***Apple*** & click on Create Customer.
v. Verify that following success message is displayed.
"Customer "Apple" has been successfully created".
vi. Logout from the application.

Test Case 2:

AT_Customer_Delete

i. Login to ***ActiTime*** application with valid username & password.
ii. Click on ***Tasks*** then click on ***Projects & Customer***.
iii. Select the checkbox of ***Apple*** customer & click on ***Delete Selected***.
iv. Click on ***Delete this Customer button*** present on confirmation popup.
v. Verify that following success message is displayed.
"***Selected customers have been successfully deleted***".
vi. Logout from the application.

Test Case 3:

AT_Billing_Create

i. Login to ***ActiTime*** application.
ii. Click on ***Settings*** then click on ***Billing Types***.
iii. Click on ***Create New Billing Type button***.
iv. Enter the name as '***contract'*** & click on ***Create Billing Type button***.
v. Verify that following success message is displayed.
"***Billing type has been successfully created.***"
vi. Logout from the application.

Test Case 4:

AT_Billing_Delete

i. Login to ***ActiTime*** application.
ii. Click on ***Settings*** then click on ***Billing Types***.
iii. Click on ***Delete link*** of contract billing type.
iv. Click ***OK*** on confirmation popup.
v. Verify that following success message is displayed.
"Billing type has been successfully deleted."
vi. Logout from the application.

With respect to above 4 sample test cases, we can identify that following steps are repeating,

i. Login
a. Enter username
b. Enter password
c. Click on Login

ii. Logout
a. Click on Logout link.

iii. Go to Customer page
a. Click on Tasks
b. Click on Projects & Customers

iv. Go to Billing page
a. Click on Settings
b. Click on Billing Types

v. Verify success message
a. Get the expected success message
b. Get the actual success message
c. Compare expected & actual success message, then Report the status (pass or fail).

All the above repeated steps are very specific to the application; hence they should be developed inside Project specific class.

While developing Project specific methods, most of the cases every method needs WebDriver instance. If we write a code to create a instance of WebDriver in every method then whenever we call these methods, it always opens the new browser.

Instead of this we should pass WebDriver instance as argument for every project specific methods, which will increase the maintenance of the code, instead of this we can create the Global variable & initialize it with the help of parameterized constructor as show below.

1. Right click on ***Libraries*** package, Go to ***New*** select ***Class*** specify the class name as ***Project Specific*** & click Finish.

2. Write the following code.

```
package com.projectname.libraries;

import org.openqa.selenium.WebDriver;

public class ProjectSpecific
{
     WebDriver driver;
     public ProjectSpecific(WebDriver driver)
     {
          this.driver=driver; //constructor
     }
}
```

```
//1. Login method
public void login(String userName, String password)
     {
          driver.findElement(By.name("username")).sendKeys(userName);
          driver.findElement(By.name("pwd")).sendKeys(password);
          driver.findElement(By.id("loginButton")).click();
     }

//2. Logout method
public void logout()
     {
          driver.findElement(By.linkText("logout")).click();
     }
//3. Go to Customer page method
public void goToCustomerPage()
     {
          driver.findElement(By.xpath("//div[text()='Tasks']")).click();
          driver.findElement(By.linkText("Projects & Customers")).click();
     }
//4. Go to Billing Page
public void goToBillingPage()
     {
          driver.findElement(By.xpath("//div[text()='Settings']")).click();
          driver.findElement(By.linkText("Billing Types")).click();
     }
//5. Verify success message
public void verifySuccessMsg(String eMsg)
     {
          String aMsg=driver.findElement(By.className("successmsg")).getText();
          Assert.assertEquals(aMsg, eMsg);
     }
}
```

Q. How do you perform verification in TestNG (Compare expected & actual Result)?

A: Using Assertions (The methods available under assert class are called as Assertions).

Note:

In ***assert class*** we have ***many over loaded assert methods*** are present & all of them are ***static***.

Ex:

```
1. assertEquals
2. assertNotEquals
3. assertTrue
4. assertFalse
5. assertnull
6. assertNotNull
7. assertSame          etc.
```

****Q. What is the difference between 'assert' and 'verify'?**

A: (This is the question related to Selenium IDE Seleness).

Both '***assert'*** & '***verify'*** functions will compare expected value with actual value, if the comparison is successful they will report the status as '***pass'*** or else they will report the status as '***fail'***.

When the comparison fails '***assert'*** will stop the execution of current test case but '***verify'*** allow us to continue the execution.

Developing SuperTestNG class:

SuperTestNG will be a ***parent class*** which contains ***BeforeMethod & AfterMethod***. These methods will perform the steps which are common for all the Test cases.

Right click on "***com.projectname.libraries***" package, Go to ***New*** & select ***class***, specify the class name as '***SuperTestNG***' & click 'Finish'.

Develop BeforeMethod & AfterMethod as shown below.

**

```
package com.projectname.libraries;

import java.util.concurrent.TimeUnit;

public class SuperTestNG
{

        public WebDriver driver;
        public ProjectSpecific pm;
        public Generic gm;
        public String URL="http://127.0.0.1/login.do";

        @BeforeMethod(groups={"all"})
        public void preCondition()
            {
                driver=new FirefoxDriver();
                Reporter.log("Opening Firefox Browser",true);
                driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
                Reporter.log("implicitlyWait is configured to 10 secs",true);
                driver.get(URL);
                Reporter.log("Navigating to URL: "+URL);
                pm=new ProjectSpecific(driver);
                gm=new Generic();
            }
        @AfterMethod(groups={"all"})
        public void postCondition()
            {
                driver.quit();
                Reporter.log("Closing the browser",true);
            }
}//End of class
```

**** Developing Automation Script:**

This is the actual stage where Manual testcases are converted into Automation Script. Generally for every Manual testcase we will create a TestNG class which has same name as the respective testcase ID & all the TestNG class will inherit from SuperTestNG class.

A TestNG class will have one test method & the name of the test method will be same as respective TestNG class name & it will have a prefix "test".

Automation Script for TestCase-1:

1. Right click on '***src'*** folder Go to ***New*** & select '***package'***, specify the package name as '***com.projectname.testscripts'*** & click '***Finish'***.

2. Right click on the above package, Go to '***New'*** & select '***class'*** specify the name as '***AT_Customer_Create***' & click 'Finish'.

3. Write the following

```
package com.projectname.testscripts;

import org.openqa.selenium.By;

public class AT_Customer_Create extends SuperTestNG
{
@Test(groups={"customer"})
public void testAT_Customer_Create()
     {
     //login to application
     pm.login("admin","manager");

     //click on Tasks & click on Project & Customers
     pm.goToCustomerPage();

     //Click on Create New Customer
     driver.findElement(By.xpath("//input[@value='Create New Customer']")).click();

     //Enter Customer Name
     driver.findElement(By.name("name")).sendKeys("Apple");

     //Click on Create Customer
     driver.findElement(By.name("CreateCustomerSubmit")).click();

     //Verify success Message
     String eMsg="Customer \"Apple\" has been successfully created.";
     pm.verifySuccessMsg(eMsg);

     //logout
     pm.logout();
     }
}//End of class
```

Automation Script for TestCase-2:

```
package com.projectname.testscripts;

import org.openqa.selenium.By;

public class AT_Customer_Delete extends SuperTestNG
{
    @Test(groups={"customer"})
    public void testAT_Customer_Delete()
    {
        pm.login("admin","manager");
        pm.goToCustomerPage();
        String xp="//tr[td[table[tbody[tr[td[a[text()='Apple']]]]]]]/td[6]/input";
        driver.findElement(By.xpath(xp)).click();
        driver.findElement(By.xpath("//input[@value='Delete Selected']")).click();
        driver.findElement(By.id("deleteButton")).click();
        String eMsg="Selected customers have successfully deleted.";
        pm.verifySuccessMsg(eMsg);
        pm.logout();
    }
}//End of class
```

Automation Script for TestCase-3:

```
package com.projectname.testscripts;

import org.openqa.selenium.By;

public class AT_Billing_Create extends SuperTestNG
{
@Test(groups={"billing"})
public void testAT_Billing_Create()
    {
    pm.login("admin", "manager");
    pm.goToBillingPage();
    driver.findElement(By.name("addLeaveType")).click();
    driver.findElement(By.id("name")).sendKeys("Contract");
    String xp="//input[contains@value.'Create Billing Type')]";
    driver.findElement(By.xpath(xp)).click();
    String eMsg="Billing type has been successfully created.";
    pm.verifySuccessMsg(eMsg);
    }
}
```

Automation Script for TestCase-4:

```
package com.projectname.testscripts;

import org.openqa.selenium.By;

public class AT_Billing_Delete extends SuperTestNG
{
    @Test(groups={"billing"})
    public void testAT_Billing_Delete()
    {
        pm.login("admin", "manager");
        pm.goToBillingPage();
        String xp="//tr[td[a[text()='Contract']]]/td[6]/a";
        driver.findElement(By.xpath(xp)).click();
        driver.switchTo().alert().accept();
        String eMsg="Billing type has been successfully deleted.";
        pm.verifySuccessMsg(eMsg);
        pm.logout();
    }
}
```

Running Automation Framework:

In order to execute all the scripts present in ***Automation Framework*** we use ***TestNG suite***.

In order to create it Right click on ***Java Project (Automation),*** Go to ***TestNG*** & select '***Convert To TestNG'***, click '***Finish'***, which creates ***testng.xml*** file inside the Java Project folder if required change the name as show below.

```
<suite name="ActiTIMESuite" parallel="none">
  <test name="Regression">
    <classes>
      <class name="com.projectname.testscripts.AT_Customer_Create"/>
      <class name="com.projectname.testscripts.AT_Customer_Delete"/>
      <class name="com.projectname.testscripts.AT_Billing_Create"/>
      <class name="com.projectname.testscripts.AT_Billing_Delete"/>
    </classes>
  </test>
</suite>
```

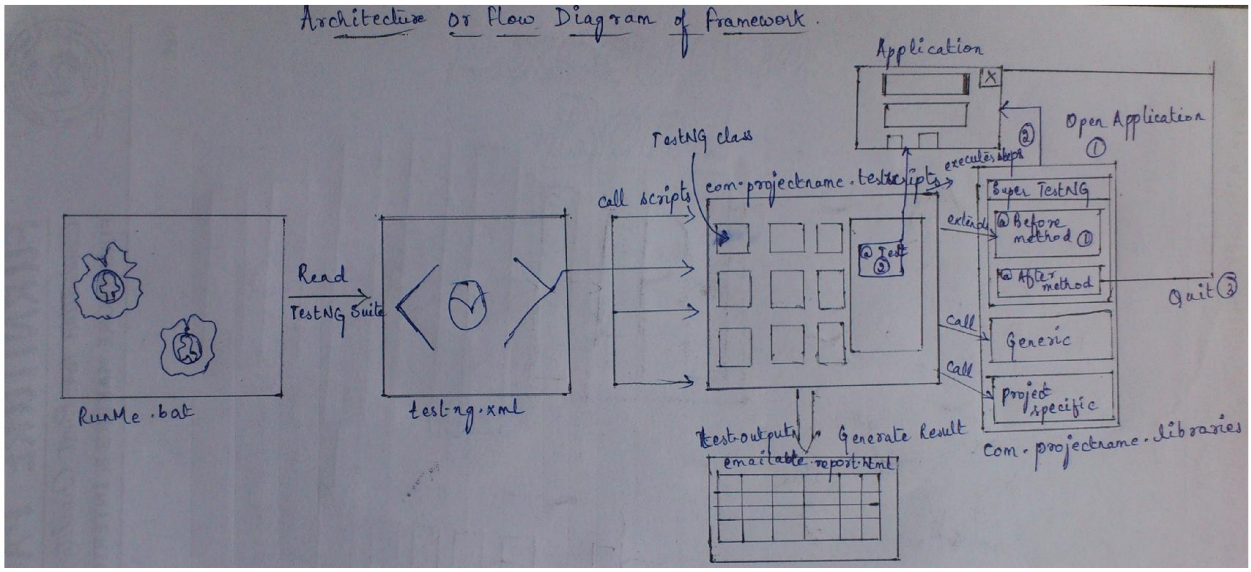
In order to execute the ***Framework***, Right click on ***testng.xml*** file, Go to ***Run as*** & select ***TestNG Suite*** after the execution, ***TestNG*** automatically generates the result in the html format which will be present inside "test-output" folder.

Running the Framework from Command prompt:

1. Go ***to Java Project*** folder (***D:\Workspace\Automation***), create a text file with the '***RunMe.bat'***.
2. Right click on the above created file & select '***Edit'***, type the following command

   ```
   java -cp bin;jars/* org.testng.TestNG testng.xml
   ```

   Save & close the file.
3. In order to execute the Framework from ***Command prompt*** double click on the above created batch file which will start the Framework execution & performs following steps.



Description:

We execute the ***Framework*** by double clicking on '***RunMe.bat***' which performs following steps.

1. It will call '***TestNG***' class which contains main method & it will pass '***testng.xml***' as argument for main method.
2. The ***testng.xml*** is a ***testng suite*** which contains list of all the testng classes which are to be executed. These ***TestNG*** classes will be present in '***com.projectname.testscript'*** package of 'bin' folder.
3. The ***TestNG class*** will execute specified testng scripts one by one & all these testng scripts are ***inherited*** from ***SuperTestNG class*** which contains ***@BeforeMethod & @AfterMethod***
4. During runtime first ***@BeforeMethod*** is executed which launches the browser, navigates to specified URL & it will also set the '***implicitlyWait timeout'***.

   After executing the test method it will execute ***@BeforeMethod*** it will execute the test method which performs the steps as per the respective ***Manual Test case***.

   After executing the test method it will execute ***@AfterMethod*** which will close all the browsers of current WebDriver instance.

5. After executing all the specified scripts ***TestNG*** will automatically generates the execution result in html format inside '***test-output***' folder of the Automation project folder.

Interview Questions for Fresher's: (Important points in CV)

Java Points:

1. Class & objects
2. Abstract & Interface
3. Constructors, Parameterized constructor, Overloaded Constructor.
4. Method Overloading, Method Overriding.
5. Inheritance & Encapsulation
6. Upcasting & Rumtime polymorphism.
7. Exception Handling
8. Collections (List & Set)
9. Database Connectivity (JDBC)
10. Threads
11. File Handling

Selenium Points:

1. Selenium IDE
2. Selenium WebDriver
3. Locators & Xpath
4. Actions class & Select class
5. Handling different types of popups such as Alert, Hidden Division and File Download etc.
6. Handling Frames
7. Performing Cross Browser testing (such as Internet Explorer & Chrome)
8. TestNG & Annotations
9. Handling Excel Files
10. AutoIT
11. Automation Framework

Q. How do you choose or select Testcase for Automation?

A: ***We select the testcase for automation based on following criteria***:

1. ***It should be Regression testcase***: This information will be provided by Manual Testing team.

2. ***It should not contain any manual interventions:***
   i. Entering the product details using barcode scanner.
   ii. Payment through credit card swipe
   iii. Notification or Alert through SMS
   iv. Bill or Invoice or Quotation printing
   v. Attendance tracking through Access card swiping
   vi. Authentication through Biometric scanners.

3. ***Tool Limitations***:
   i. Virtual keyboard in login page of www.icicibank.com
   ii. CAPTCHA (Completely Automated Public Turing test to tell Computers and Human Apart).
   iii. Verification of Audio or video clips are not possible (YouTube)
   iv. Verification of Graphical charts & Animations.

   Note: Because of above reason 100% Automation is not possible.

Q. What type of testcases or testing we don't automate?

i. Adhoc testing
ii. Usability testing
iii. Game testing
iv. Any testing which involves external hardware.

Q. What is Build? How do you receive the Build?

A: ***Build*** is a complied & compressed copy of software. We receive the Build at a predefined interval which is called as ***Build cycle*** or ***Test cycle*** such as,

i. Daily Build
ii. Weekly Build
iii. Monthly Build etc.

If the Build installation is simple then the Development team will copy paste the build on a centralized machine (Shared Folder) & the Testing team will copy the build to Testing Server & they will install it. If the Build has ***'setup.exe'*** file we will double click it & we will follow the instruction given in the installation wizard.

If the ***setup.exe*** is not available then we will manually copy paste all the required files & folders & we create a database etc. This is called as ***Build Deployment***.

If the installation & configuration is complicated then Development team or Build team itself will install the Build & they will share the URL of the new Build through email.

Q. What is Test Bed?

A: It is the ***ready testing environment*** i.e. it is a system where all the required software's are installed, Build is deployed, testdata is created & then system is ready for testcase or test script execution.

Q. What is ANT? What is the use?

A: **ANT** stands for '***Another Neat Tool'***. It is Build tool.

Developer uses ANT to create a Build.

i. Open the ***Eclipse IDE***, go to ***File→*** select '***Export'***, expand '***General'*** & select '***ANT Build Files***' click ***Next***, select the '***Java Project***' & click '***Finish'*** which creates ***Build.xml file*** inside ***Java Project***.

The ***build.xml*** contains following xml tag.

a. ***<property>***: Which is used to declare the variables.
b. ***<path>***: It is used to specify the path of the files which are to be included while creating the build .
c. ***<target>***: It is used to define the task which should be done by the ANT, it is also called as ***ANT job***.

The ANT job generally includes following steps:

i. Deleting old files & folders
ii. Creating New Folders
iii. Compiling the source code
iv. Archiving the compiled code (such as creating jar files)

Q. How do you integrate your Automation Framework with the Build?

A: Using '***Jenkins'*** which is ***a continuous integration tool***.

i. Go to following website: jenkins-ci.org, Go to '***Long term support release' tab***, click on ***older but stable link*** & click '***OK'*** which will save ***Jenkins the war file*** in the Downloads folder, copy paste the file into the required location such as ***D drive***.

ii. Open the command prompt (cmd), go to ***D drive***.

iii. Type the following command ***'java –jar jenkins.war'*** & hit enter.

If Jenkins server is loaded successfully it will display following message in command prompt.

```
'Jenkins is fully up and running'
```

iv. Open the browser & enter the following URL in the address bar "local host:8080" which will display homepage of the Jenkins server.

v. Click on ***New Job*** & specify the job name
Ex: ***Automation***, select the first radio button '***Build a free software Project'***, click ***OK***.

vi. Click on '***Advanced'*** & select last checkbox under '***Advanced Project options***' section (***use custom workspace***) & type the following path in the directory (The java project path – ***D:\Workspace\Automation***)

vii. Click on '***Add Build step'*** & select '***Execute Windows Batch Command'*** & type '***RunMe.bat'*** in the command field.
Note: In the same Jenkins job developers will select '***Invoke ANT option'*** under '***Add Build step'*** which will call the ***ANT*** & ***ANT*** will create the ***build***. The Automation team will add "***Execute windows Batch command***' option as second build step so that once the build is created the Framework execution starts automatically.
Click ***Apply*** & ***save*** which creates a ***Jenkins Job***.

viii. In order to create a ***Build***, Developer clicks on '***Build Now'*** which will start the Build creation process once the build is created it will start Framework execution automatically. Every time when we run the build, Jenkins will display a ***link*** under ***Build History***, the name of the link will be current ***system Date & Tim***e, when we click that link it will take us to Build details page where if we click on '***Console output'*** link it will display output of the Automation Framework which is printed on the command prompt.

Maven:

Maven is a ***dependency tool*** i.e. each time when the browser gets updated we need to use latest version of selenium jar file instead of manually downloading the latest version of jar file & associate with the ***Java Project***, we can do this automatically using the ***Dependency tool that is Maven***.

Installing Maven Plugin of Eclipse:

i. ***Open*** the ***Eclipse IDE***, go to ***Help*** & select ***Eclipse Market Place***, search for ***Maven***, click on ***Install button*** of ***Maven***, click '***Next'***, select ***I Accept*** & click ***Finish***. Click ***Ok*** for warning message & click '***Yes'*** which will ***restart*** the ***Eclipse IDE***.

ii. Right click on the ***Java Project***, go to ***Configure***, it should display ***Convert to Maven Project option***.

Note: Similarly install ***Maven Integration for Eclipse WTP (Juno)*** plugin for Eclipse IDE.

Converting Java Project into Maven Project:

1. Right click on ***Java Project***, go to ***Configure*** & select ***Convert to Maven Project*** & click ***Finish***, which creates ***pom.xml*** file (***Project Object Model***).

2. Double click on '***pom.xml'***, go to ***Dependencies tab***, click on ***Add***, enter ***Group ID***, ***Artifact ID*** & the ***version*** and click ***OK***

3. We can get all these information from the following URL http://docs.seleniumhq.org/download/maven.jsp

Note:
The above configuration makes the ***Maven*** to always download 2.40.0 version selenium jar file. In order to download any jar file where the version is 2.40.0 & greater we should specify following syntax in the version field.

```
  [2.40.0,)
  |       |
min      max

[2.40.0, 2.42.0] --> It downloads only 2.40.0 or 2.42.0
(2.40.0, 2.48.0) --> Range from 2.40.0 to 2.48.0
```

Q. What is CAPTCHA? How do you automate it using selenium?

A: ***CAPTCHA*** stands for (***Completely Automated Public Turing test to tell Computers and Human Apart***).

CAPTCHA is intentionally designed to avoid ***Automation***. In order to continue automating the application we can use following solutions.

1. Fix the ***CAPTCHA*** for every build received for testing where we hardcode the value of the ***CAPTCHA*** in selenium script.

2. Add the ***Attribute*** for the ***CAPTCHA*** which will have the ***CAPTCHA value***, so that we can retrieve it using ***getAttribute*** & enter in the application.

Q. What is the process you follow for Automation?

A: Explain Automation Test Life Cycle.

Automation Test Life Cycle is a step by step procedure which is followed to Automate testing of the application, which involves following stages,

1. **System Study:** In this phase the Automation team will refer the manual test cases instead of requirement document. They will read the testcases & they will identify whether the testcase can be automated or not i.e. the test case should be regression testcase, it should contain any manual intervention & tool should be able to automate it.

2. **Test Plan:** It is a document which contains future testing activities, in most of the cases we do not create separate plan for automation, we update Test Automation section of Manual Test Plan itself with the following information.
   i. Automation tool
   ii. ***Licensing***: Ex- ***GPL*** (General Public License)
   iii. Automation team size
   iv. Roles & Responsibilities of each Automation Engineer
   v. Server or system Information
   vi. Help documents related to Automation Framework, Libraries & coding standards etc.
   vii. Schedule that is start & ending time of Automation task
   viii. Standard template

3. **Develop Test Script:** This is the actual stage where identified Manual testcases are converted into Automation scripts.

   In order to convert Manual testcases into Automation scripts, we follow below mentioned steps,

   a. Read the Testcase & execute it manually at least once so that it gives more clarity on the steps which are to be automated.
      Develop the methods for repeating steps.

   b. Write the script as per Manual testcase steps.
   c. Execute the script in the local machine & ensure that it is working. Send it for approval.
   d. Store it in Test Script Repository after the approval.

Q. How do you review the Automation Script & how do you provide review comments?

A: While reviewing the Automation Script we consider following points.

a. Ensure that script is running without any errors or exception.
b. Ensure that all the steps of Manual Testcase are automated.
c. Ensure that Methods are developed instead of repeating codes.
d. Ensure that code is written with proper indentation (tab space).
e. Ensure that java naming conventions are followed.
f. Ensure that inline comments are provided wherever it is necessary.

After reviewing the script we provide the review comment in the review comment template as shown below.

Sl No	ScriptName	Line	Comment	Severity	Reviewer	Author	AuthorComment	Date
1	AT_Customer_Create	5	no try-catch block	High	Bhanu	Keshav		

Traceability Matrix:

It is a document which is used to map the testcase against the requirement to ensure that every requirement has at least one testcase. The same document is also used to track Automation status.

ReqID	TC_ID1	Automation
Req1	TC1	Yes
Req2	TC2	No
Req3	TC3	NA

In the existing ***Traceability Matrix*** document we add a column called "***Automation***" & we specify anyone of the following values in the Automation column.

Yes: It indicates the testcase is Automated.

No: It indicates the testcase is not yet automated.

NA: It indicates the testcase will not be automated, it could be because it may not a ***Regression testcase*** or testcase may include ***manual intervention***.

In order to communicate Automation status to the Manual testing team we use Traceability Matrix so that Manual testing team will come to know whether to execute the testcase manually or whether it will be taken care by Automation team. Using Traceability Matrix document we will also come to know other information such as *Total number of testcases automated, total number of testcases which are pending for Automation, percentage of Automation coverage etc.*

Framework Execution and Analyzing Results:

When we receive a build we will install it in the lab machine & we start the Framework execution by double clicking on the batch file or we can do this automatically using ***continuous integration tool*** such as ***Jenkins***. After the execution of the framework we will analyze the result for failures.

When a script is failed we execute respective testcase manually, if the manual testcase itself is failed it indicates that there is a bug in the application, in such cases we will submit the defect in the ***Defect Tracking Tool***.

If the Manual testcase is '***Passed'*** it indicates that there is a bug in the Automation script, in such cases we ***debug the script.*** After analyzing all the results we prepare ***Script Execution Report*** as shown below & then we will publish it.

Sl No	Script_ID	Status	Comment
1	AT_Customer_Create	Pass	NC
2	AT_Customer_Delete	Pass	NC
3	AT_Billing_Create	Fail	Bug123
4	AT_Billing_Delete	Fail	NullPointerException

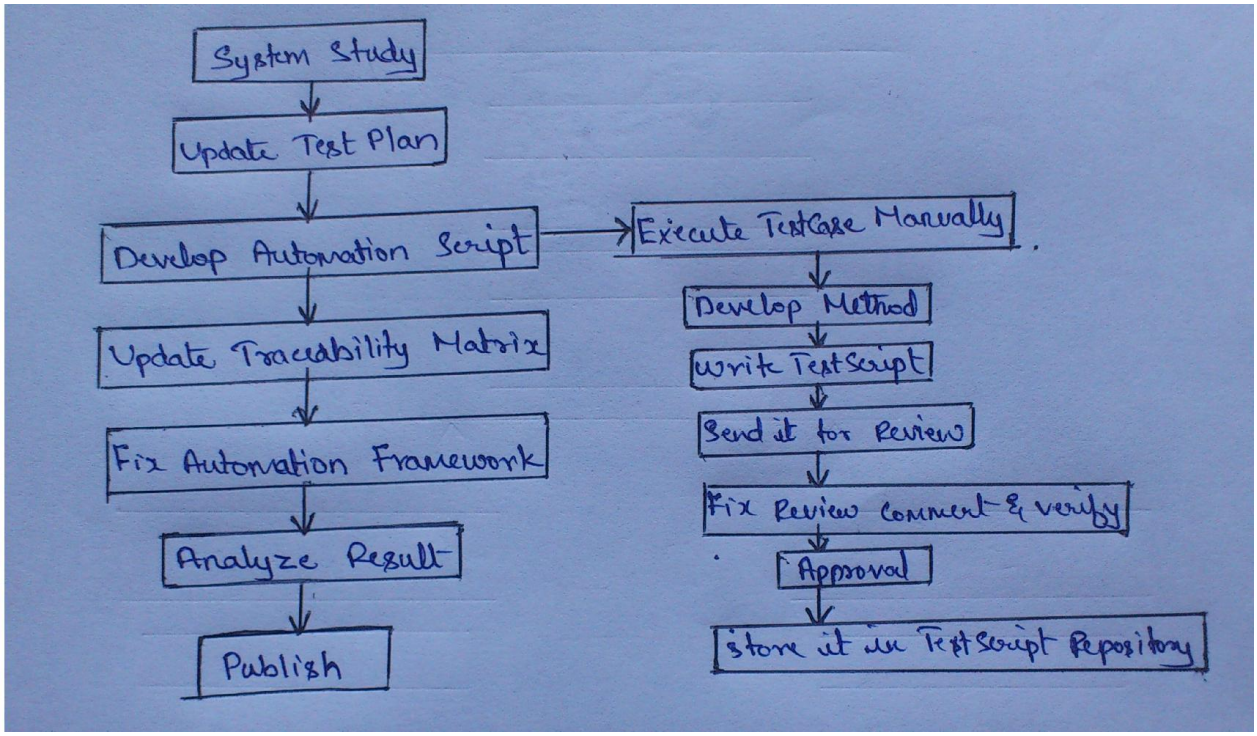
Q. What is Debugging and how do you debug the code?

A: Process of executing the code line by line or step by step in order to detect the bug is called ***Debugging***.

In order to debug the code we should perform following steps.

a. Place the cursor in the required line, go to Run & select '***Toggle Breakpoint'***.
b. To start the execution in the debugging mode press **F11.**
c. In order to execute the code step by step press **F6.**
Note: In order to debug the method use **F5.**

Automation Test Life Cycle Diagram:



Installing the build:

When we receive the build, if it contains setup.exe file, then we double click on the file & we will follow the instructions provided on Installation wizard & it performs following steps.

a. Creates the required folder
b. It will copy paste the files
c. It will create the database & the tables
d. It will configure the application
e. It creates the shortcut on the Desktop

```
Ex: ActiTIMEsetup.exe
```

Build Deployment:

In this scenario we should manually perform all the above mentioned steps, such as Creating folders, copying, creating database & tables etc.

While deploying the build we have 2 major steps

1. **Install the server**
   Ex: Double click on ***WAMP server exe file*** (***Window Apache MySQL PHP***), click ***Next***, select ***I Accept***, click ***Next***, click ***Next***, select the ***Checkboxes***, click ***Next***, click ***Install***, click ***Next*** & click ***Finish***.

2. **Deploying the Build:**
   ```
   Ex: Bamboo invoice
   ```
   a. Right click on ***the build file (zip file)*** & select ***extract here*** which will unzip the build.
   b. Copy the ***Bamboo invoice folder*** & paste it inside ***C:\wamp\www***
   c. Go to following location
      ```
      C:\wamp\www\bamboinvoice\bambo_system_files\application\config
      ```
   d. Open ***'database.php'*** in ***WordPad*** & scroll down completely & remove the password.
   ```
   $db['default']['password']='root';
   $db['default']['password']='';
   ```
      Save the file & ***close it***.
   e. Specify the ***'bamboinvoice'*** in ***create new database field***, then click on ***create***.
   f. Close & reopen the browser & go to ***local host***.
   g. Click on ***bamboinvoic***e which is available under '***Your Project'***
   h. Click on 'You can Install it Now'
   i. Enter the following information

Email:	
Primary contact:	
Password:	
Password Confirm:	
Install	

   j. Close & reopen the browser, go to following URL: http://localhost/bamboinvoice/index.php/login

Selenium Grid:

In order to execute the script on ***Remote machine***, we use ***Selenium Grid***. It will have ***2 machines***.

1. **Hub**: It is a ***server machine*** where the automation script is present (***local machine***)

2. **Node**: This is the C***lient machine*** or the ***Remote machine*** on which we want to execute the automation script.

   Before executing the script on Remote machine, we need to start the Hub, then we need to start the Node, which will interact or register with the Hub, then we should run the script.

Starting the Hub:

1. Go to the ***local machine***, ***download*** & ***copy paste*** the ***selenium server jar file*** into the required location.
   `Ex: C:\Drive, if required renamte it (eg: sss.jar)`
2. Open the ***command prompt*** & type the following command,
   `C:\>java -jar sss.jar -role hub`

Starting the Node:

Note:

To start the ***Node*** we should know the ***IP address*** of the ***server machine***, in order to get it, go to the server machine (Hub), open ***command prompt***, type ***ipconfig*** & hit ***Enter*** which will display the ***ip*** address. `Ex: 192.168.1.22`

1. Go to Client machine, download & copy paste Selenium jar file into required location
   `Ex: D:\Drive, rename it if it is required.`
2. In the Client machine, open command prompt & type D: & hit enter.
   `D:\>java -jar sss.jar -role node -hub http://192.168.1.22 :4444/grid/register`
   & it runs in port 5555.

Executing the Script:

Go to Hub machine, open the Eclipse & write the following code

```
package com.qspiders;

import java.net.MalformedURLException;
import java.net.URL;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

public class Demo
{
    public static void main(String[] args) throws MalformedURLException
    {
    URL hubURL=new URL("http://192.168.1.22:4444/wd/hub");
    DesiredCapabilities browser = DesiredCapabilities.firefox();
    WebDriver driver=new RemoteWebDriver(hubURL,browser);
    driver.get("http://www.google.com");
    }
}
```