Selenium Web driver

Steps to configure Eclipse:

1. Create a workplace (folder) in 'D' drive.
2. Select the workplace, while launching Eclipse & click on "Launch".
3. After launching click on open perspective & select java & click on open.
4. After that create a "Java project".

   Note: Do not create module info.
5. Right click on 'Src' folder → Select new → Select package

   Note: package name always starts with lowercase.
6. Right click on package → new → Select class

   Note: class name always starts with upper case.

Types of Applications:

Generally – 2 types: 1. Internet application
2. Intranet application

Internet Application:

The applications which requires internet & everyone can use is called internet application.

Intranet Application:

The applications which can be used only by some particular people is called as Internet application.

Note: If we divide the applications, we are having 4 types:

1. Standalone
2. Mobile
3. Distributed
4. web

Stand-alone application:

- The applications which doesnot requires internet is called stand-alone application.
- To develop stand-alone applications we need J2SE (Java 2 Standard Edition)

Ex: Ms-Excel, Ms powerpoint, Notepad etc;

Mobile Applications:

- The applications which we can use in a mobile is called Mobile App.

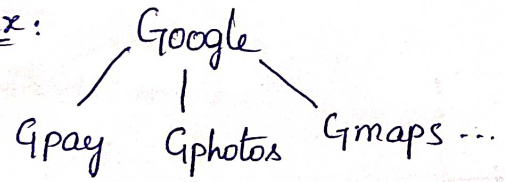
Ex: Calculator, pubg

- To develop mobile applications we need to use J2ME (Java 2 Micro Edition) with flutter technology.

Distributed Application:

- When the main app. is distributing it's properties to the sub-applications with the help of cloud technology is called distributed application.

Ex:



Web Application:

- The applications which require internet is called web applications.

→ Web app. having 2 ends: 1. Front end 2. Backend.

- To develop front end, we have to use web tech. (HTML, CSS, Javascript).
- To develop back end, we need 2 things 1. J2EE with database.

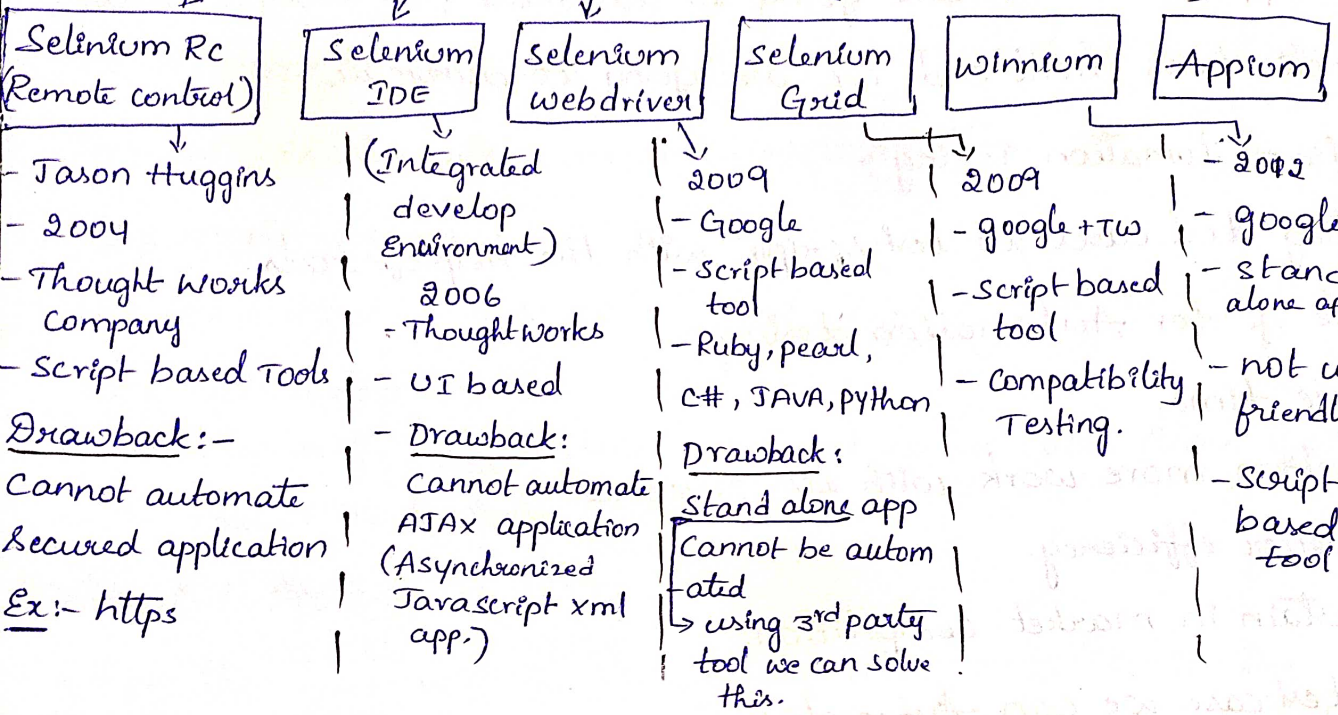
Note:

- To connect J2EE with database we have to use JDBC (Java Database Connection).
- To connect Front end with backend we have to use servlets.

Automation Testing

1. Which Application we are going to automate?
   - Web applications front end we are going to automate.
2. What is Automation Testing?
   - Converting test cases to test scripts with the help of tools.
3. Why we go for Automation testing?
   - To save time.
   - To perform more work with less resource.
   - For better efficiency.
   - To sustain in market competition.
4. Which test case we can Automate?
   - Regression test cases can be automated.
5. When we can go for Automation testing?
   - We can go for Automation testing if the application is stable.
6. Which test cases cannot be automated?
   - OTP related, Captcha related, Barcode related Testcases.
   - If manual testcases are incorrect.
   - Gaming related, MP3, MP4 related Testcases
7. What are the automation tools we are having?
   - Selenium (open source tool)
   - QTP (Quick Test professional) [Not an open source tool, license based tool].
8. What are the 3rd party tools we are having?
   - Auto IT
   - Sekuli.

History of Selenium



* The father of selenium is "Jason Huggins".
* Selenium is a community which consists of Automation testing tools.

Selenium Rc:

- Stands for Selenium Remote control
- Script based Automation testing tool, developed in the year 2004 by thought works company.
- The drawback of the tool is, it cannot automate secured applications.

Selenium IDE:

- It is an UI based Automation Testing tool developed in the year 2006 by thought works company.
- The drawback of this tool is it cannot automate AJAX applications (Asynchronized javascript XML application).

Selenium Webdriver:

- It is a script based application automation tool developed in the year 2009 by google.
- It is a most successful automation testing tool.
- It supports programming languages like Ruby, pearl, C#, java, python.
- The only drawback of this tool is, it cannot automate stand alone application directly (with the help of 3rd party tool we can automat

Selenium grid:

- It is also a script based automation testing tool developed in the year 2009 by google + Thoughtworks.
- It is same as selenium webdriver, only one extra feature is there, that it can perform compatibility testing.

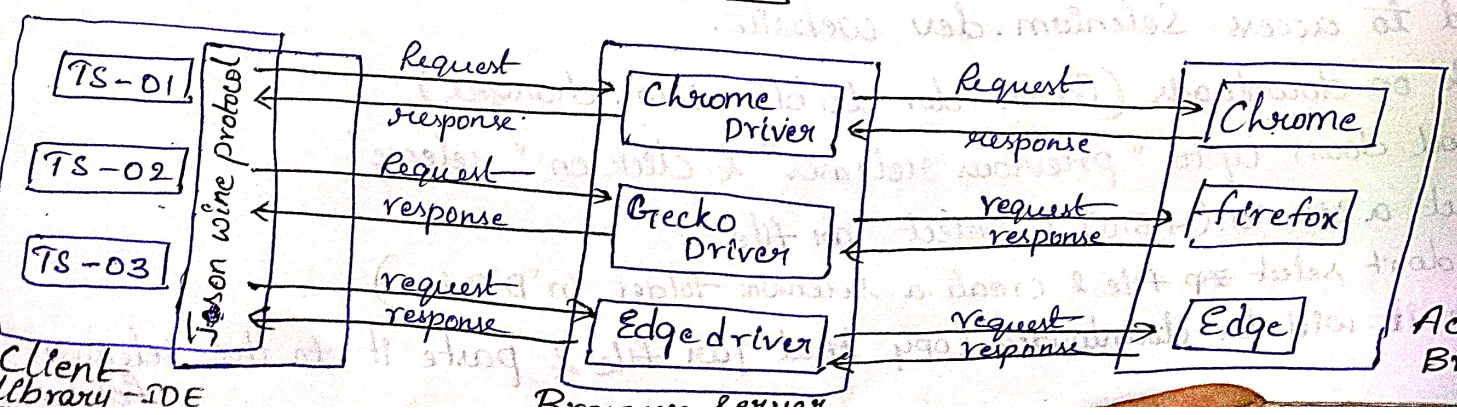
Winnium:

- It is also a script based automation testing tool developed in the year 2012 by google.
- This tool can automate stand alone app. directly but this tool is not successfull because it is not user friendly.

Appium:

- It is also a script based Automation testing tool developed by source labs in the year 2014.
- It is generally used to automate mobile app.

Communication level Architecture:



- In case of Selenium webdriver Communication level architecture 3 major participants are. (6)

1, Client library 2, Browser server 3, Actual ~~ser~~ Browser.

Client library:

- Client library is nothing the collection of inbuilt classes & interfaces, by using those inbuilt classes & interfaces, we are going to write test scripts.
- Inside client library Jason wire protocol is present, which will convert java to JSON & JSON to Java (JSON :- Javascript object Notation).

Browser Server:

- Brocoser server is a place where all the drivers are present.
  Ex: chrome driver, Gecko driver, Edge driver.
- The functionality of drivers are to carry the request to the actual browser & carry back the response to the client library.

Actual Browser:

- These are nothing but stand alone app. present in our system. inside this actual browser we are going to perform the execution.

Configuration of Selenium webdriver:

1, Selenium webdriver Jar file
2, JDK 1.8
3, Browser server
4, Actual Browser.

Steps to download Selenium Jar file:

- Need to access Selenium.dev website.
- Click on downloads (Green color to blue color changed)
  Scroll down upto "previous releases" & click on "release"
  Select a version (4.5.0) & select jar file.
  (Note: don't select zip file & create a selenium folder in 'D' Drive)
  Once it will be downloaded copy that jar file & paste it to the selenium folder.

Steps to add Jar file in our project:

→ Right click on project → select build path → configure build path

↓

Libraries ← classpath ← add external jars ← Select jar file from selenium folder

(Don't click on module path)

↓

Apply & close (Note: Jar file will be visible inside reference libraries).

Steps to download chrome Driver:

Go to Selenium.dev website → downloads → Scroll down & click on "Browsers"

↓

Chrome Browser Documentation ← Select latest stable release (Based on chrome browser version) ← chromedriver_win32.zip

(Note: Browser version & driver version has to be seen)

After downloading steps:

- Once we have downloaded, we need to extract to selenium folder. (Note: don't select extract here)
- Select either extract all / extract files.

Web driver and web Element methods:

1. get ("url");
   - with the help of this method we can access to a web application.
   Syntax: driver.get ("url");
2. getTitle ();
   - with the help of this method we can get the current web page title.
   Syntax: driver.getTitle ();
3. getPageSource ();
   - with the help of this method we can fetch the front end source code of the current web page.
   Syntax: driver.getPageSource ();
4. getCurrentUrl ();
   - with the help of this method we can fetch the url of current webpage.
   Syntax: driver.getCurrentUrl ();
5. Maximize ();
   - This method will help to maximize the current window.
   Syntax: driver.manage ().window ().maximize ();
6. Minimize ();
   - This method / operation / minimization included from selenium 4, in selenium 3 we cannot perform minimization operation.
   - This method will help to minimize the window.
   Syntax: driver.manage ().window ().minimize ();
7. Set Size ();
   - With the help of this method we can customize the window dimensions.

Syntax: driver.manage().window().setSize(new Dimension(0,0));

(or)

```
Dimension d = new Dimension(0,0);
driver.manage().window().setSize(d);
```


8. setposition();

- with the help of this method, we can relocate the windowsw.

Syntax: driver.manage().window().setposition(newpoint(0,0));

(or)

```
point p = new point(0,0);
driver.manage().window().setposition(p);
```

9. close();

- with the help of this method, we can close the parent (superclass) window.

Syntax: driver.close();

10. findElement();

- with the help of this method, we can identify a single web element
- Return type is web element.

Syntax: driver.findElement(By.locator("string"));

11. findElements():

- with the help of this method, we can identify multiple web elements.
- Return type is List<webelement>

Syntax: driver.findElements(By.locator("string"));

12. Sendkeys():

- with the help of this method, we can send text value to the text box in a web page.

Syntax: Element.sendkeys("string");

13, click():

- will help us to perform clicking operation in a web page.

Syntax: element.click();

Launching chrome Browser:


```
package
 /* step 6 : import . chrome Driver */
import org . openqa . selenium . chrome . ChromeDriver;
 /* step 3 : import . chrome option */
import org . openqa . selenium . chrome . chromeoptions;
public class program1
{
  psvm (string [] args)
  {
    /* step 2 : create object ~~for~~ for chromeoptions class */
     Chrome options option = new ChromeOptions ();
     option . add Arguments ("--remote-allow-origins=*");
   /* step 4 : provide key & value for chrome Driver */
System . setproperty ("webdriver . chrome . driver", " D:\\ selenium version \\
   /* step 5 : create an object for chrome Driver */      Chromedriver.exe");
     ChromeDriver driver = new Chrome Driver (option);
   /* step 7 : use get method to access web app. */
     driver . get ("https://www . flipkart . com/*");
  }
}
```

Conclusion:

- we are using chrome options to customise the chrome driver session. & it will handle [remote http connection failed exception]
- we have to use a method c'd as set-property to provide the key & value of chrome drive.

[Note: if we donot provide it properly, we will get (illegal state exception)]

- We have to use a method c'd get to access to the web application.

Steps to download Geckodriver:

- Go to Selenium.dev website → downloads → scroll down & select 'Browsers'

↓

Firefox Documentation → Geckodriver releases → From 0.33 select Zip file (32.zip (or) 64.zip).

After download: → extract the Zip file to selenium folder.

Scenario 1:

- Access Actitime.com
- Fetch the current web page title, url & source code.

⇒ program ↓

```
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
Public class program2
{
  Psvm (String[] args)
  {
    ChromeOptions.option = new ChromeOptions();
    option.addArguments ("--remote_allow_origins=*");
    System.setproperty ("webdriver.chrome.driver", "D:\\Selenium folder\\
    Chromedriver.exe");
    Chrome Driver driver = new chrome Driver(option);
    driver.get ("https://demo.actitime.com/login.do");
    String Title = driver.getTitle();
    String Src code = driver.getPage Source();
    String url = driver.get current Url();
    Sopln (Title);
    Sopln (Src code);
    Sopln (url);
```

Maximize, minimize, Resize, Relocate operations in a current window.

Scenario

1. Launch chrome browser
2. Access actitime
3. Maximize the window
4. Minimize the window
5. Again maximize the window
6. Resize the window
7. Relocate the window
8. close the window.

```
import org.openqa.selenium.Dimension;
import org.openqa.selenium.point;
import org.openqa.selenium.chrome.chromeDriver;
import org.openqa.selenium.chrome.chromeoptions;
public class program3
{
  Psvm (string[] args)
  {
    Chromeoptions option = new Chromeoption();
    option.addArguments("--remote_allow-origins=*");
    System.setproperty("webdriver.chrome.driver", "D://selenium folder//chrome driver.exe");
    chromeDriver.driver = new chromeDriver();
    driver.get("https://demo.actitime.com/login.do");
    Thread.sleep(4000);
    driver.manage().window().maximize();
    Thread.sleep(4000);
    driver.manage().window().minimize();
    Thread.sleep(4000);
    driver.manage().window().maximize();
    Thread.sleep(4000);
    /* Approach 1 */
// driver.manage().window().setsize(new Dimension(0,0));
/* Approach 2 */
// Dimension d = new Dimension(0,0);
```

```
        driver.manage().window().setSize(d);
        Thread.sleep(4000);
      /* approach 1 */
    // driver.manage().window().setPosition(new Point(0,0));
      /* approach 2 */
  // Point p = new Point(0,0);
    driver.manage().setPoint(P);
    Thread.sleep(4000);
    driver.close();
  }
}
```

Scenario 2:

1. Launch chrome Browser
2. Maximize the window
3. Resize the window
4. Relocate the window
5. Again maximize the window
6. Fetch the current webpage url
7. Minimize the window
8. Again maximize & close the windows.

Note: in between every operation, 3 seconds time interval.

```
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOption;
public class program
{
  psvm (String[] args)
  {
    ChromeOptions option = new ChromeOptions();
    option.addArguments("--remote_allow_origins=*");
System.setProperty("webdriver.chrome.driver","D://Selenium folder//
Chrome Driver.driver");
    ChromeDriver driver = new ChromeDriver(option);
```

```
driver.get("hHps://www.flipkart.com");
Thread.sleep(3000);
driver.manage().window().maximize();
Thread.sleep(3000);
driver.manage().window().setSize(newDimension(400,500));
Thread.sleep(3000);
driver.manage().window().setposition(new point(500,500));
Thread.sleep(3000);
driver.manage().window().maximize();
Thread.sleep(3000);
String Title = driver.getTitle();
String url = driver.getCurrentUrl();
Thread.sleep(3000);
driver.manage().window().minimize();
Thread.sleep(3000);
driver.manage().window().maximize();
//Sopln(Title);
//Sopln(url);
 driver.close();
 }
}
```

Performing Access, Back, Forward, Refresh operations with the help of Navigation interface & Navigate method.

To perform all these operations, we have to use some methods which are present inside Navigation inteface.

1. to("url");

Syntax: driver.navigate().to("url");

2, back();

will help us to go back to the previous web page.

Syntax: driver.navigate().back();

3, forward();

will help us to go to the next web page

Syntax: driver.navigate().forward();

4, refresh();

will do refresh the current web page

Syntax: driver.navigate().refresh();

Scenario:

1, Launch chrome Browser
2, Access actitime.com
3, Access flipkart
4, Go back to actitime
5, Go forward to flipkart
6, Refresh the current web page
7, in between every operation, 3 sec time interval.

⇒
```
import org.openqa.selenium.webdriver.Navigation;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

public class Back_fwd_refresh_operations
{
    psvm (String[] args)
    {
        ChromeOptions option = new ChromeOptions();
        option.addArguments("--remote-allow-origins=*");
System.setProperty("webdriver.chrome.driver", "D://Selenium folder// chrome driver.exe");
        ChromeDriver driver = new ChromeDriver();

        /* Approach 1 */

    Navigation n = driver.navigate();
```

```
// driver.get ("https://demo.actitime.com/login.do");
    n.to ("https://demo.actitime.com/login.do");
driver.manage().window().maximize();
Thread.sleep(4000);
    n.to ("https://www.flipkart.com/");
// driver.get ("https://www.flipkart.com/");
Thread.sleep(4000);
n.back();
Thread.sleep(4000);
n.forward();
Thread.sleep(4000);
n.refresh();

/* Approach 2 */

driver.navigate().to("https://demo.actitime.com/login.do");
driver.manage().window().maximize();
Thread.sleep(4000);
driver.navigate().to("https://www.flipkart.com/");
Thread.sleep(4000);
driver.navigate().back();
Thread.sleep(4000);
driver.navigate().forward();
Thread.sleep(4000);
driver.navigate().refresh();

}
}
```

Scenario:

- Launch chrome browser
- Access myntra.com
- Access Ajio.com
- Maximise the window
- Minimize the window
- Again maximise
- Go back to previous webpage
- Go forward to next page
- Resize & Relocate the window
- Maximise the window
- Refresh the window
- Fetch the title & url
- close the windows.

Note: in between every operation, 4 sec time interval.

⇒

(18)

```
class program
{
  psvm (string [] args)
  {
    ChromeOptions option = new ChromeOptions ();
    option.addArguments ("--remote-allow-origins=*");
System.setProperty ("webdriver.chrome.driver","D://selenium folder//chromedriver.exe");
    ChromeDriver driver = new ChromeDriver (option);
    driver.get ("https://www.myntra.com");
    Thread.sleep (4000);
    driver.get ("https://www.Ajio.com");
    Thread.sleep (4000);
  driver.manage ().window ().maximize ();
   Thread.sleep (4000);
  driver.manage ().window ().minimize ();
   Thread.sleep (4000);
  driver.manage ().window ().maximize ();
  Thread.sleep (4000);
  driver.navigate ().back ();
  Thread.sleep (4000);
 driver.navigate ().forward ();
 Thread.sleep (4000);
```

Scenario:

- Launch chrome browser
- Access myntra.com
- Access Ajio.com
- Maximize the window
- Minimize the window
- Again maximize
- Go back to previous webpage
- Go forward to next page
- Resize & Relocate the window
- Maximize the window
- Refresh the window
- Fetch the title & url
- close the window.

Note: in between every operation, 4 sec time interval

⇒


```
class program
{
  psvm (string [] args)
  {
    ChromeOptions option = new ChromeOptions ();
    option.addArguments ("--remote-allow-origins=*");
System.setProperty ("webdriver.chrome.driver","D://selenium folder//chromedriver.exe");
    ChromeDriver driver = new ChromeDriver (option);
    driver.get ("https://www.myntra.com");
    Thread.sleep (4000);
    driver.get ("https://www.Ajio.com");
    Thread.sleep (4000);
  driver.manage().window().maximize();
    Thread.sleep (4000);
  driver.manage().window().minimize();
    Thread.sleep (4000);
  driver.manage().window().maximize();
    Thread.sleep (4000);
  driver.navigate().back();
    Thread.sleep (4000);
  driver.navigate().forward();
  Thread.sleep (4000);
```

```
driver.manage().window().setSize(new Dimension(400,500));
Thread.sleep(4000);
driver.manage().window().setPosition(new point(500,600));
Thread.sleep(4000);
driver.manage().window().maximize();
Thread.sleep(4000);
driver.navigate().refresh();
Thread.sleep(4000);
String Title = driver.getTitle();
Sopln(Title);
String Url = driver.getCurrentUrl();
Sopln(url);
driver.close();
}
}
```


HTML

- HTML Stands for Hyper Text Mark up language.
- It is not a case-sensitive programming language.
- In case of HTML, 3 major things are
  1. Tag name 2. Attribute 3. Visible text.

Tag name:

The value which is present ~~inside~~ just after the '<' & bracket is nothing but called as Tagname.

Example: <input> ; <a> ;

Attribute:

- The information which is present inside the tag is called as Attribute.
- All attributes must contain some value, which is considered as attribute value.

Example: <input type = "text">

Note: multiple attributes can be present inside a tag.

Visible text:

The text value which is present in between same open and close tag is consider as visible text.

Example: Login

Rules of HTML:

- HTML page starts with <html> tag.
- After <html> tag, we should provide <head> tag.
- Once we close <head> tag, we need to provide <body> tag.

Note: <body> tag will not comes under <head> tag.

Example:

```
<html>
    <head>
    </head>
    <body>
    </body>
</html>
```

Heading ← WELCOME

Username [ ]

Password [ ]

Radio Buttons
- ○ Male
- ○ Female
- ○ others

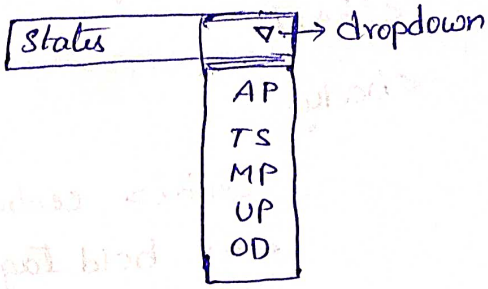
Checkbox ← ☐ Accept terms & conditions

button ← [Submit]

```
<html>
  <head>
    <title> Registration </title>
  </head>
  <body>
    <centre> centre tag
    <b> bold tag
    <h1> Welcome </h1> Heading tag
    <lable> Username </lable>
    <input type = 'text'>
    <br> </br> break tag
    <lable> password </lable>
    <input type = 'password'>
    <br> </br>
    <input type = 'radio'>
    <lable> male </lable>
    <br> </br>
    <input type = 'radio'>
    <lable> female </lable>
    <br> </br>
    <input type = 'radio'>
    <lable> Others </lable>
    <br> </br>
    <input type = 'checkbox'>
    <lable> Accept terms & conditions </lable>
    <br> </br>
    <input type = 'button' value = 'submit'>
    <br> </br>
    </br>
    </center>
</html>
```

Creation of Dropdown:

```
<html>
    <head>
        <title> dropdown </title>
    </head>
<body>
        <lable> states </lable>
        <select>
        <option> AP </option>
        <option > TS </option>
        <option > MP </option>
        <option> UP </option>
        <option > OD </option>
</body> </select>
</html>
```



Student Registration:

First name
Middle name
Last name
user name
password
State

O Male
O Female
O Others
☐ Trust all the content.

Submit

```
<html>
    <head>
        <title> Student Registration
                                </title>
    </head>
<body>
    <center>
    <b> <h1> Student Registration </h1>
    < lable> First name </lable>
    <input type = "text">
    <br> </br>
    < lable> Middle name </lable>
    <input type= "text">
                    <br></br>
    < lable> Last name </lable>
    <input type = "text">
    <br> </br>
```

```
<lable> password </lable>
<input type = 'text'>
<br> </br>
<lable> States </lable>
<select>
<option> Ap <option>
<option> MP <option>
<option> TS <option>
<option> K <option>
</select>
<input type = 'radio'>
<lable> male </lable>
<br> </br>
<input type = 'radio'>
<lable> female </lable>
<br> </br>
<input type = 'radio'>
<lable> others </lable>
<br> </br>
<input type = 'checkbox'>
<lable> Trust all the content </lable>
<br> </br>
<input type = 'button' value = 'submit'>
</b>
</centre>
</body>
</html>
```

Web table :

```
<html>
  <head>
    <title> web page creation </title>
  </head>
<body>
  <h3> student details </h3>
  < table border = "1">
    < tbody >
      <tr> (table row)
        <td > name </td> ( table data)
        <td> Roll No </td>
        <td> age </td>
      </tr>
      <tr>
        <td> Aniruddha </td>
        <td> 1 </td>
        <td> 100 </td>
      </tr>
      <tr>
        <td> Abhishek </td>
        <td> 2 </td>
        <td> 99 </td>
      </tr>
  </tbody>
  </ table>
  <h3> Book details </h3>
  < table border = "1">
  < tbody >
      <tr>
        <td> Bookname </td>
        <td> Authorname </td>
        <td> book price </td>
      </tr>
      <tr>
        <td> Harry potter </td>
        <td> J.K. Rowling </td>
        <td> 1000 </td>
      </tr>
      <tr>
        <td> Ramayanam </td>
        <td> valmiki </td>
        <td> 500 </td>
      </tr>
  </tbody>
  </table>
  </body>
  </html>
```

Locators

Web element:

The elements which are present in a web page is called as web element.

Locators:

- Locators are static methods present in 'By' class.
- Locators are used to locate the web element present in the web page.
- we have 8 types of locators.

1. id()
2. name()
3. class name()
4. link Text()
5. partial Link Text()
6. css Selector()
7. Xpath()
8. tagname()

id():

- Will work only with id attribute.
- Search syntax in web page:

  [id = 'idAttribute value']

  Eg: [id = 'username'] (Example)

- Syntax in Script:

```
driver.findElement(By.id("id attribute value"));

driver.findElement(By.id("username")); (Example)
```

name():

will work only with name attribute.

Search syntax in webpage:

[name = 'name Attribute value']

Eg: [name = 'pwd'] (Example)

Script Syntax:

driver.findElement(By.name("name attribute"));

driver.findElement(By.name("pwd")); (Example)

className():

will work only with class Attribute.

Search syntax in webpage:

[class = 'class attribute name']

[class = 'initial'] (Example)

Script Syntax:

driver.findElement(By.className("class attribute value"));

driver.findElement(By.className("initial")); Example)

Scenario:

- Launch chrome Browser
- Access actitime
- perform login operation with the help of id, name, classname locator:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
Public class LoginActiTime
{
  Psvm (String [] args) throws Exception
  {
    ChromeOptions option = new ChromeOptions();
    option.addArguments("--remote-allow-origins=*");
System.setProperty("webdriver.chrome.driver", "D:\\selenium folder\\chrome driver.exe");
    ChromeDriver driver = new ChromeDriver(option);
    driver.get("https://demo.actitime.com/login.do");
  driver.manage().window().maximize();
    Thread.sleep(3000);
  WebElement element = driver.findElement(By.id("username"));
    element.sendkeys("admin");
    Thread.sleep(3000);
WebElement element1 = driver.findElement(By.name("Pwd"));
    element1.sendkeys("manager");
    Thread.sleep(3000);
WebElement element2 = driver.findElement(By.className("initial"));
    element2.click();
                    (or)
// driver.findElement(By.id("username")).sendkeys("admin");
// driver.findElement(By.name("Pwd")).sendkeys("manager");
// driver.findElement(By.className("initial")).click();
  }
}
```

CSS Selector ( ):

~~cssSe~~ css Selector will work with all the attributes as well as css expressions.

Search Syntax in web page:

tagname [attribute name = 'Attribute value']

input [Placeholder = 'Username'] (example)

Script Syntax:

driver. findElement (By. ~~cssSe~~ cssSelector ("tagname [attribute name . 'attribute value']"));

driver. findElement (By. cssSelector ("input [Placeholder . 'username']")); (example)

Special features of css:

Note 1:

(dot) . is the substitute of class in case of cssselector.

Example: input . textField . pwd-field
↘ class

Note 2:

(hash) # is the substitute of id in case of cssSelector.

Example: input # username
↘ id

Scenario:

- Launch chrome Browser
- Access actitime
- Do the login operation with the help of cssSelector.

```
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
public class LoginActitime_cssSelector
{
  Psvm (String[]args) throws Exception
  {
    ChromeOptions option = new ChromeOptions();
    option.addArguments("--remote-allow-origins=*");
System.setProperty("webdriver.chrome.driver","D:\\seleniumfolder\\chrome
                                                    driver.exe");
    ChromeDriver driver = new ChromeDriver(option);
    driver.get("https://demo.actitime.com/login.do");
    driver.manage().window().maximize();
    Thread.sleep(4000);
    driver.findElement(By.cssSelector("input[Placeholder='username']"))
                                                    .sendkeys("admin");
    driver.findElement(By.cssSelector("input[Placeholder='password']")).
                                                    .sendkeys("manager");
    Thread.sleep(4000);
    driver.findElement(By.cssSelector("a[id='loginButton']")).click();
}}
```

Scenario:

- Launch chrome Browser
- Access OrangeHRM.com
- do the login operation for Orange HRM Application.

⇒

```
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
Public class LoginActitime_cssSelector
{
   Psvm (String[]args) throws Exception
    {
      ChromeOptions option = new ChromeOptions();
      option.addArguments("--remote-allow-origins=*");
System.setproperty("webdriver.chrome.driver","D:\\Selenium folder\\chrome
                                                    driver.exe");
      ChromeDriver driver = new ChromeDriver(option);
      driver.get("https://demo.actitime.com/login.do");
      driver.manage().window().maximize();
      Thread.sleep(4000);
      driver.findElement(By.cssSelector("input[Placeholder='username']"))
                                                    .sendkeys("admin");
    driver.findElement(By.cssselector("input[Placeholder='password']")).
    Thread.sleep(4000);                             .sendkeys("manager");
    driver.findElement(By.cssSelector("a[id='loginButton']")).click();
} }
```

Scenario:

- Launch chrome Browser
- Access OrangeHRM.com
- do the login operation for OrangeHRM Application.

Scenario:

- Launch chrome Browser
- Access Flipkart
- close the login page
- In the search text box, search iphone 11 & click on the search button
- close the window

⇒

```
import org.openqa.selenium.By;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
class program
{
  Psvm (—>
   {
      ChromeOptions option = new ChromeOptions();
      option.addArguments("--remote-allow-origins=*");
System.setproperty("webdriver.chrome.driver", "D:\\Selenium folder\\chromedriver.exe")
      ChromeDriver driver = new ChromeDriver(option);
      driver.get("https://www.flipkart.com/");
driver.findElement(By.cssSelector("button[class='_2KpZ6l _2doB4z']"))
Thread.sleep(4000);
                                          .click();
driver.findElement(By.cssSelector("input[placeholder='Search for products,
                    brands and more']")).sendkeys("iphone 11");
Thread.sleep(4000);
driver.findElement(By.cssSelector("button[class='L0Z3Pu']")).click();
Thread.sleep(4000);
driver.close();
}
}
```

Scenario:

- Launch chrome Browser
- Access actitime
- Do the login operation
- Maximize the window
- click on task, Reports, users
- click on Time track
- close the window.

⇒

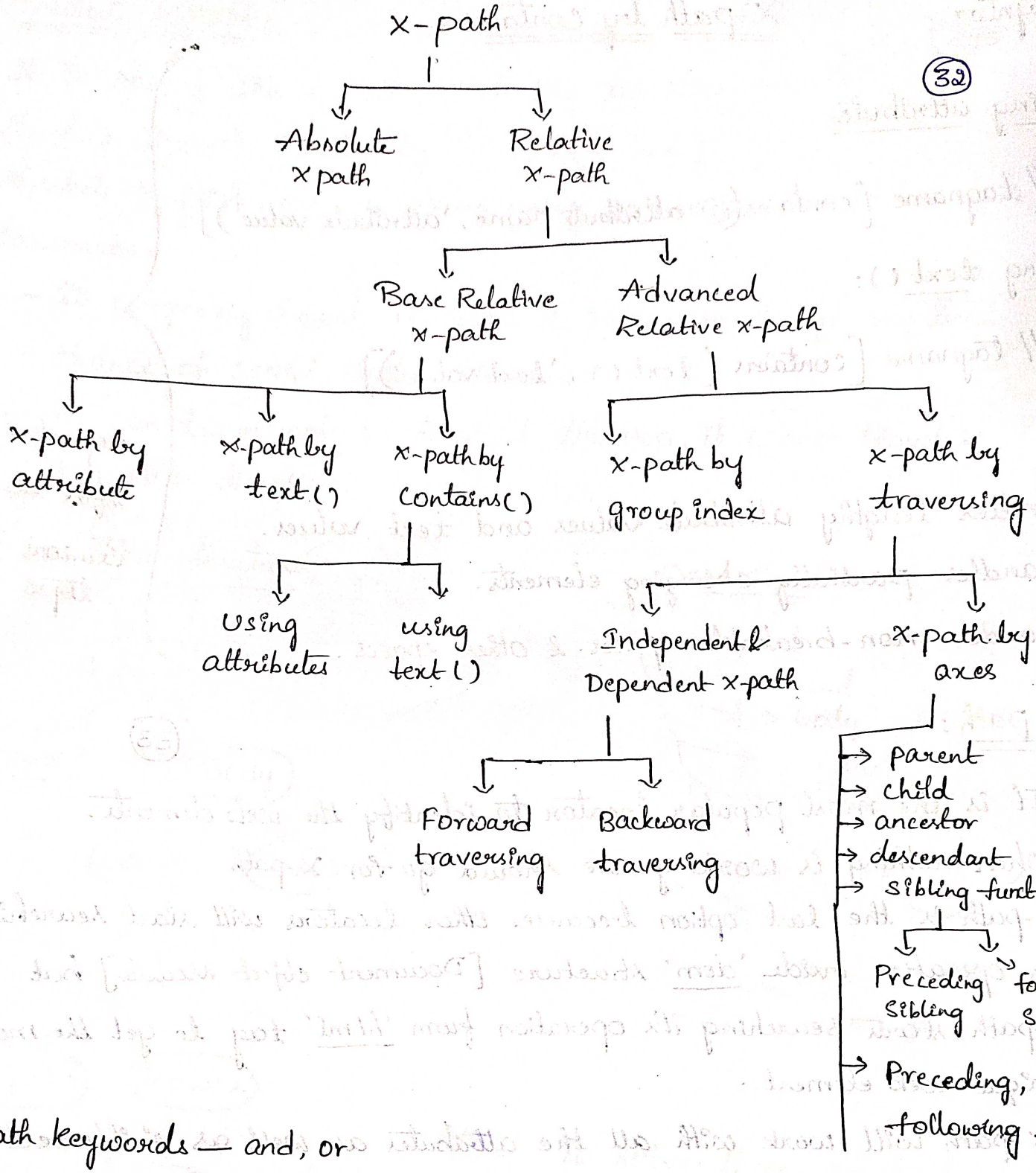
```
Public class Actitime_clickonModules
{
    ChromeOptions option = new ChromeOptions();
    option.addArguments("--remote-allow-origins=*");
System.setProperty("webdriver.chrome.driver", "C:\\selenium version\\chromedriver.exe");
    ChromeDriver driver = new ChromeDriver(option);
    driver.get("https://demo.actitime.com/login.do");
    Thread.sleep(4000);
    driver.findElement(By.cssSelector("input[placeholder='username']")).SendKeys("admin");
    Thread.sleep(4000);
    driver.findElement(By.cssselector("input[Placeholder='Password']")).sendkeys("manager");
    Thread.sleep(4000);
    driver.findElement(By.id("loginButton")).click();
    Thread.sleep(4000);
    driver.manage().window().maximize();
    Thread.sleep(3000);
    driver.findElement(By.id("container_tasks")).click();
    Thread.sleep(3000);
    driver.findElement(By.id("container_reports")).click();
    Thread.sleep(3000);
    driver.findElement(By.id("container_users")).click();
    Thread.sleep(3000);
    driver.findElement(By.id("container_tt")).click();
    Thread.sleep(3000);
    driver.close();
}
```

Scenario:

- Launch chrome Browser
- Access actitime
- Do the login operation
- Maximize the window
- click on task, Reports, users
- click on Time track
- close the window.

⇒

```
Public class Actitime_clickonModules
{
    ChromeOptions option = new ChromeOptions();
    option.addArguments("-remote_allow-origins=*");
System.setProperty("webdriver.chrome.driver", "C:\\selenium version\\chromedriver.exe");
    ChromeDriver driver = new ChromeDriver(option);
    driver.get("https://demo.actitime.com/login.do");
    Thread.sleep(4000);
    driver.findElement(By.cssSelector("input[placeholder='username']")).SendKeys("admin");
    Thread.sleep(4000);
    driver.findElement(By.cssselector("input[Placeholder='Password']")).sendkeys("manager");
    Thread.sleep(4000);
    driver.findElement(By.id("loginButton")).click();
    Thread.sleep(4000);
    driver.manage().window().maximize();
    Thread.sleep(3000);
    driver.findElement(By.id("container_tasks")).click();
    Thread.sleep(3000);
    driver.findElement(By.id("container_reports")).click();
    Thread.sleep(3000);
    driver.findElement(By.id("container_users")).click();
    Thread.sleep(3000);
    driver.findElement(By.id("container_tt")).click();
Thread.sleep(3000);
driver.close();
}
```



X-path keywords — and, or

X-path functions:

1. text()
2. contains()
3. name()
4. starts with()
5. normalize space()
6. last()

X-path:

- It is the most popular locator to identify the web elements.
- When nothing is working, we should go for x-path.
- x-path is the last option because other locators will start searching it's operation inside 'dom' structure [Document object Module] but x-path starts searching it's operation from 'html' tag to get the more unique web element.
- x-path will work with all the attributes as well as visible text & link text.

Types of x-path:

2 types:

1. Absolute x-path
2. Relative x-path

Absolute x-path:

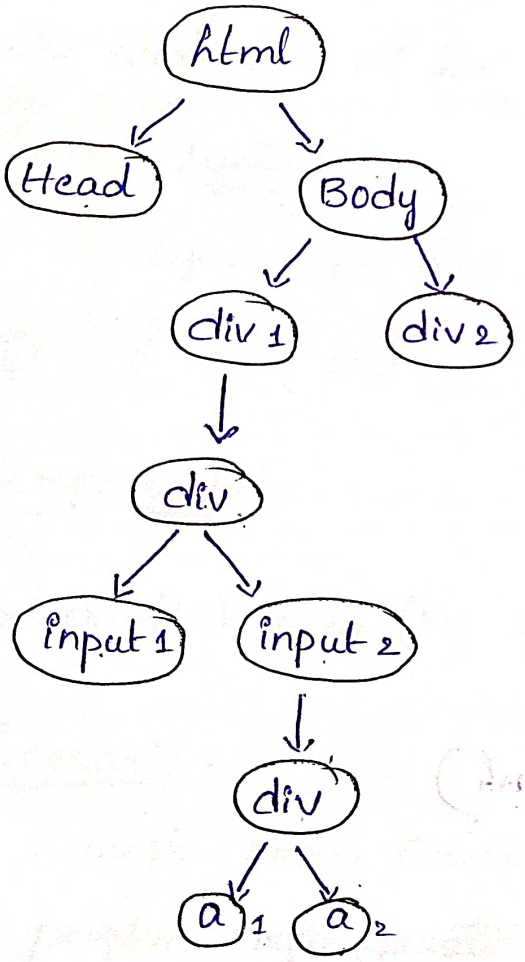
→ It is one of the x-path to identify the web elements.

→ Absolute x-path denotes by '/' (single slash)

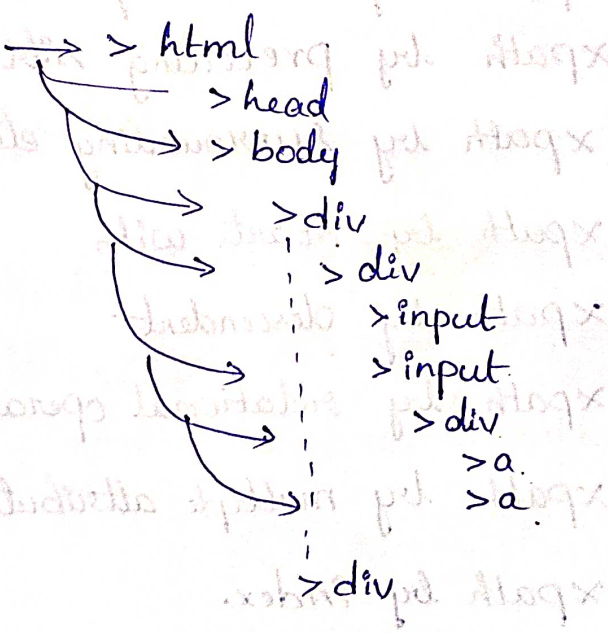
→ Absolute x-path, we are never going to use because it has some drawbacks.

- It completely depends on index & index can change any time.
- Chance of script failure is more.
- it can travel only in forward direction. it cannot travel in any other direction.

Ex: Tree structure



DOM structure



Note: To solve these drawbacks, you have to go for relative x-path.

Syntax: Eg:

/html/body/div[1]/div/input[2]/div/a[2]

Relative x-path:

Basic Relative x-path:

- xpath by attribute index.
- xpath by visible Text.
- xpath by contains attribute value.
- xpath by contains visible value.

Advance Relative x-path:

- xpath by forward traversing.
- xpath by backward traversing.
- xpath by following sibling.
- xpath by preceding sibling.
- xpath by surrounding elements.
- xpath by starts with
- xpath by descendent
- xpath by relational operator.
- xpath by multiple attributes & visible text.
- xpath by index.

Note:

Relative x-path denotes by '//' (double slash)

1. xpath by attribute value:

will work with all the attributes.

Search Syntax:

//tagname [@attribute name = 'attribute value']

Ex: // input [@ placeholder = 'username']

Script syntax:

Eg:

driver.findElement(By.xpath("//input[@placeholder='username'])).sendkeys("Admin");

2. xpath by visible Text:

will work with all the visible text & link text

Search syntax:

//tagname [text() = 'visible text']

Ex: // div [text() = 'Login']

Script syntax:

Eg:

driver.findElement(By.xpath("/div[text()='login']")).click();

Scenario:

- Launch chrome Browser
- perform login operation of Actitime with the help of x-path.

```
class program
{
  Psvm (____)
   {
     chrome options option = new chromeoptions();
     Option. addArguments ("--remote-allow-origins=*");
System.setproperty ("webdriver.chrome.driver", "D:\\selenium folder\\
                                                chromedriver.exe");
     chrome Driver = new chrome Driver (option);
     driver. get ("https://demo.Actitime.com/login.do");
     driver. manage(). window(). maximize();
driver. findElement (By.xpath ("//input [@placeholder='username']")).
  Thread. sleep (4000);                    Sendkeys ("admin");
driver. find Element (By.xpath ("//input [@placeholder='password']")).
  Thread. sleep (4000);                    Sendkeys ("manage");
  driver. find element (By.xpath ("// div[text()='Login']")). click();
  driver. close();
}
}
```

X-path Keywords: and, or

X path functions:

1. text();
2. Contains();
3. name();
4. starts with()
5. normalize-space()
6. last()

X-path by contains():-

- X path returns directly to an element using either attribute (or) Tex

Syntax:

Using attributes:

// tagname [contains (@attributename, 'attribute value')]

Using text():

// tagname [contains (text(), 'text value')]

Uses:

- Handles lengthy attribute values & text values.
- Handles partially change in elements.
- Handles Non-breakable spaces & other spaces also.

Advanced xpath Group index:

Advanced Relative x-path:

xpath by Group index:

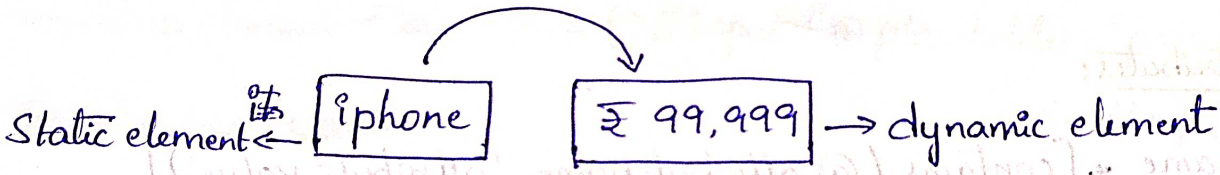
Whenever we have multiple matching elements, we go with xpath by group index.

Syntax: (xpath expression) [Position value]

Eg: (//a [contains (text(), 'features')]) [1]

Xpath by traversing:

Steps to write xpath of completely dynamic element:



1. Identify the static element and write xpath.
2. Identify common parent.
3. Write either tagname | tagname with attributes of dynamic element.

Independent & dependent xpath:

Independent xpath:

xpath of a static element is called Independent xpath

Dependent xpath:

xpath of dynamic element is called dependent xpath.

- we have 2 types under Independent & dependent xpath:

1. Forward traversing
2. Backward traversing.

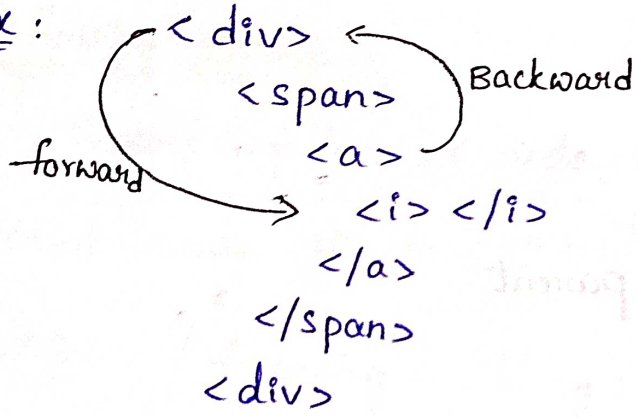
Forward traversing:

Traversing from a node to it's immediate child node using forward slash (/) is called forward traversing.

Backward traversing:

Traversing from a node to it's immediate parent node using forward slash (/..) is called backward traversing.

Ex:



forward

//div/span/a/i

Backward

//a/../..

Assignment

Icc - cricket.com website → click on view full rankings → Ratings of virat kohli

Bollymovie reviewz.com → Box office collection → xpath of verdict of any movie

Demoapp.skillrary.com → Course → selenium Training → xpath for price

Xpath by Axes:

1. Parent Axes:

Traversing from a node to it's immediate parent.

Syntax: /parent :: tagname

2. Child Axes:

Traversing from a node to it's immediate child.

Syntax: /child :: tagname

3. ancestor axes:

Traversing from a node to any parent

Syntax: /ancestor :: tagname

4. descendant axes:

Traversing from a node to any child.

Syntax: /descendant :: tagname

Assignment:

Olympics.com → Scroll down till featured Athletes → xpath of ① (Novak Djokovic)

Sibling Functions:

1, Preceding sibling:

Traversing from a node to the above nodes which are at the same level and having common parent.

Syntax: /preceding-sibling :: tagname

Hyphen.

2, Following sibling:

Traversing from a node to the nodes which are below the current node at the same level having common parent.

Syntax: /following - sibling :: tagname

Preceding:

Traversing from a node to the nodes lying above the current node which are at the same level having uncommon parents.

Syntax: /preceding :: tagname

Following:

Traversing from a node to the nodes lying below the current nodes which are at the same level having uncommon parents.

Syntax: /following :: tagname.

Xpath Functions:

1. Last function:

It is used to fetch the last element of the list of elements

Syntax: // tagname [last()]

2. normalize-space():

It is used to ignore unnecessary spaces in the attribute values or text values including non breakable spaces.

For Attributes: // tagname [normalize-space (@Attributename = 'Avalue')]

For text: // tagname [normalize-space (text() = 'TAV')]

3. Starts-with():

It fetches all the elements which starts with specified attribute value (or) text value. It accepts 2 arguements.

Syntax:

for attributes: //tagname [starts-with (@AN, 'AV')]

for text: // tagname [starts-with (text(), 'TAV')]

4. name():

It is used to fetch the elements whose tag name is not identifiable using normal xpath.

Ex:- <svg> tag

Syntax: // * [name() = 'svg'] [@AN = 'AV']

Xpath keywords:

Whenever we wanted to pass multiple attributes (or) text functi-ons (or) Contains functions (or) combination of any of this combination Then we use xpath keywords.

1. //tagname [@AN$_1$ = 'AV$_1$' and/or @AN$_2$ = 'AV$_2$']
2. //tagname [@AN = 'AV' and/or text() = 'TV']
3. //tagname [@AN = 'AV' and/or contains (,)
4. //tagname [text() = 'TV$_1$' and/or text() = 'TV$_2$']
5. //tagname [text() = 'TV' and/or contains (,)]
6. //tagname [contains (,) and/or contains (,)]

Link text locator:

- It is used to identify a link using the text present on it.
- Links are identified using <a> (anchor) tag.

Syntax: linkText ("<text on the link>")

WebElement link = driver.findElement (By.linkText ("<text on the link>"));

Partial linkText locator:

- It is used to identify a link uniquely using partial text on the link.
- We go for partial link text locator when the text on the link is lengthy.
- It is also used to handle partially dynamic elements

Syntax:

WebElement link = driver.findElement (By.partialLinkText ("<partial text on the link>"));

tagname locator:

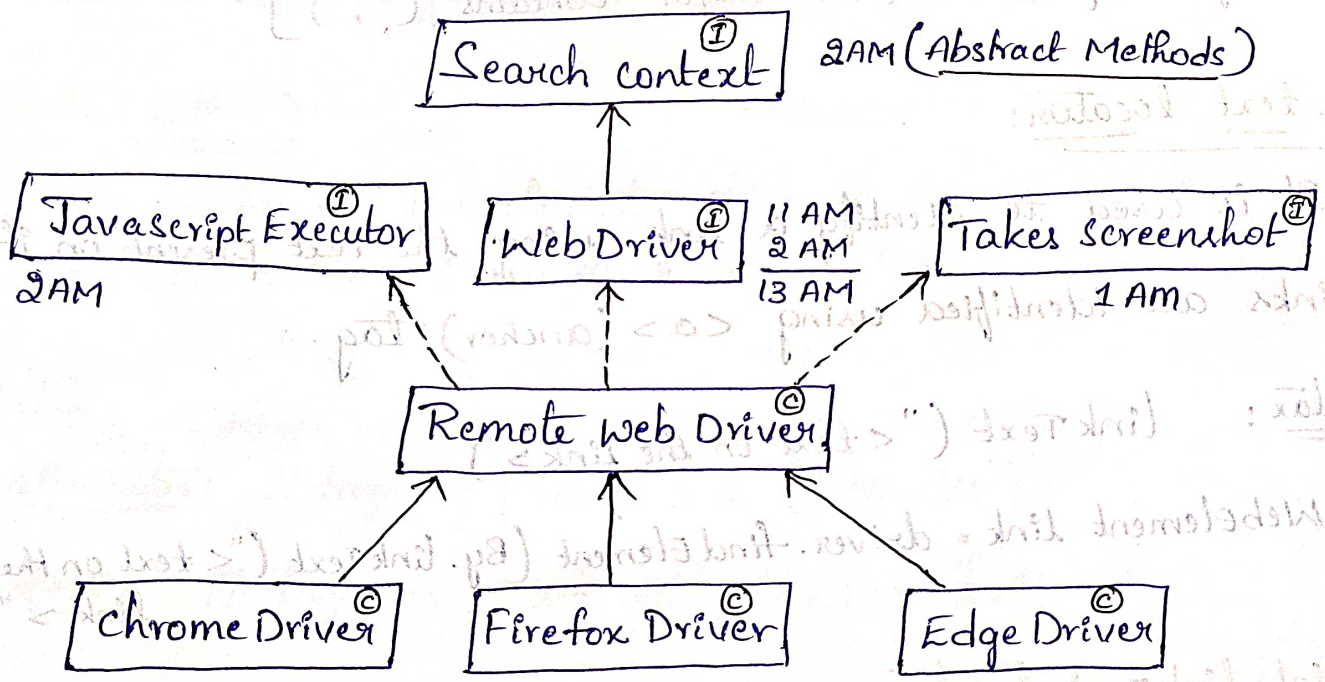
It is used to identify an element (or) list of elements using tagname.

Syntax:

```
WebElement e = driver.findElement(By.tagName("<tagname>"));

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

WebDriver's class Diagram (or) Java selenium Architecture / selenium WebDriver Architecture w.r.to Java



- Search context is the supermost interface in selenium WebDriver class diagram. It is used to search element/elements in the web page. It has 2 Abstract methods 1, findElement() & 2, findElements().

- WebDriver interface is the sub interface of search context interface. It performs all driver related actions on the web page.

It has 11 Abstract methods of it's own, 2 Abstract methods inherited from search context, all together it has 13 Abstract methods.

1. get()
2. getTitle()
3. getCurrentUrl()
4. getPageSource()
5. getWindowHandle()
6. getWindowHandles()
7. manage()
8. navigate()
9. switchTo()
10. close()
11. quit()
12. findElement() } inherited from
13. findElements() } Search context.

(46)

- RemoteWebDriver is the implementing class of WebDriver interface. It also implements Javascript executor interface and TakesScreenshot interface.
- JavascriptExecutor interface is used to execute javascript code in Selenium. It has 2 Abstract methods.
  1. executeScript()
  2. executeAsyncScript()
- TakesScreenshot interface is used to capture the screenshot of the webpage. It has 1 Abstract method
  1. getScreenshotAs()
- All the browser specific classes like ChromeDriver, FirefoxDriver, EdgeDriver etc, extends to RemoteWebDriver class.

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

- The above statement is used to launch empty chrome Browser
- With respect to Java, the above statement is the upcasting statement.

- We are creating an instance of browser specific class and upcasting into WebDriver interface.

- The above statement is an example for Run time polymorphism, based on the object created, respective browser is launched.

Synchronization

- It is the process of matching selenium speed with Application speed.

Why? Synchronization

Application takes more time to load compared to selenium program which might cause exceptions such as

- No such element Exception
- Time Out Exception
- Element Not Interactable Exception

etc;

To avoid these exceptions, we go for synchronization.

→ we have different types of Wait statements:

1. Thread.sleep()
2. ImplicitlyWait()
3. Explicitly wait
   - WebDriverWait ©
   - Fluent Wait ©
4. Custom wait

1, Thread.sleep():

It is Java wait statement which waits completely for the specified amount of time.

- It accepts arguments of the data type long.
- It accepts time in milliseconds.
- It throws InterruptedException.

2, implicitlyWait():

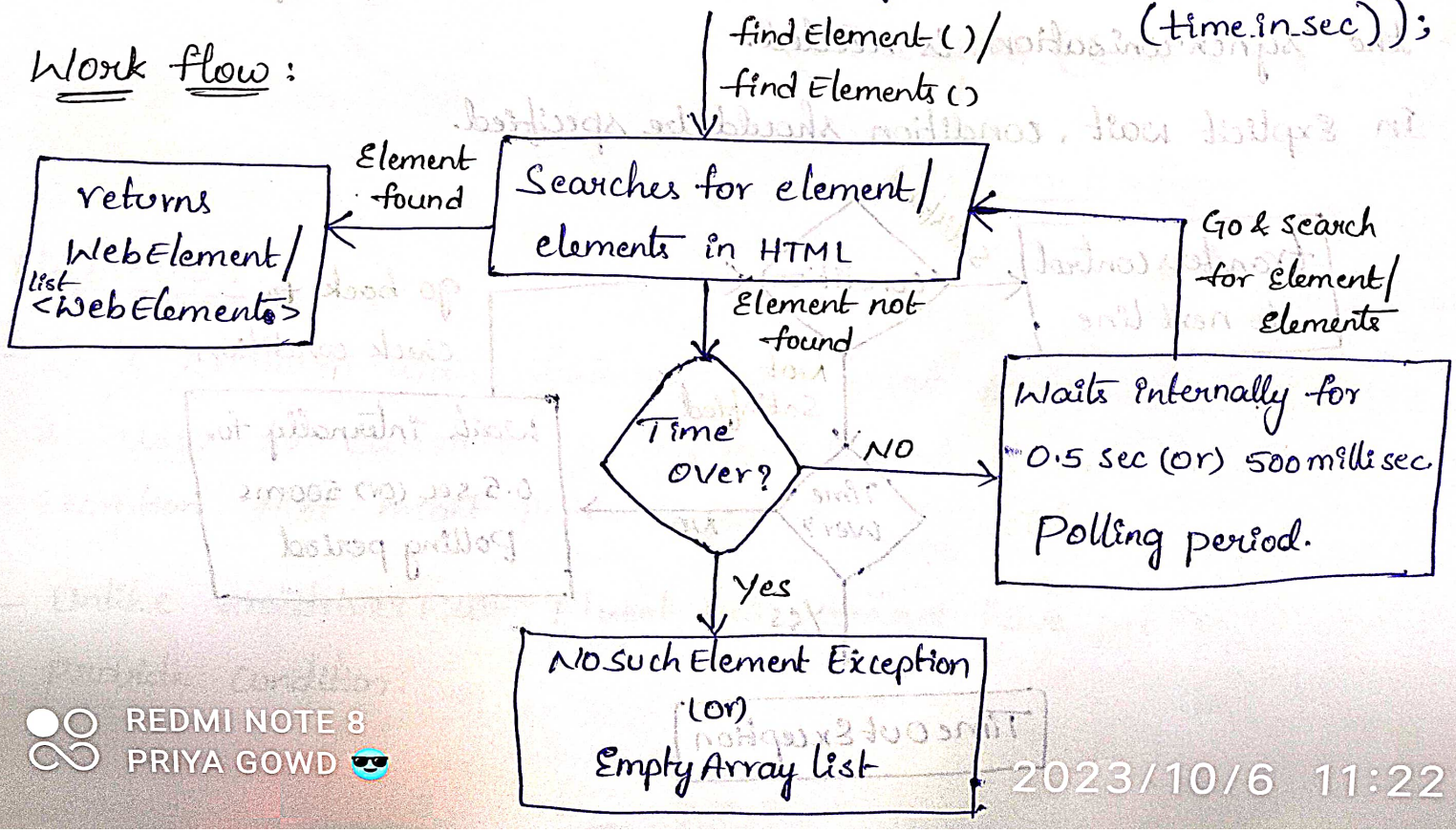
It is selinium wait statement which synchronizes findElement and findElements() methods only.

- It is the global wait statement which can be given at the beginning of the script and it synchronizes all the findElement & findElements methods throughout the script.

Syntax:

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(time_in_sec));

Work flow:

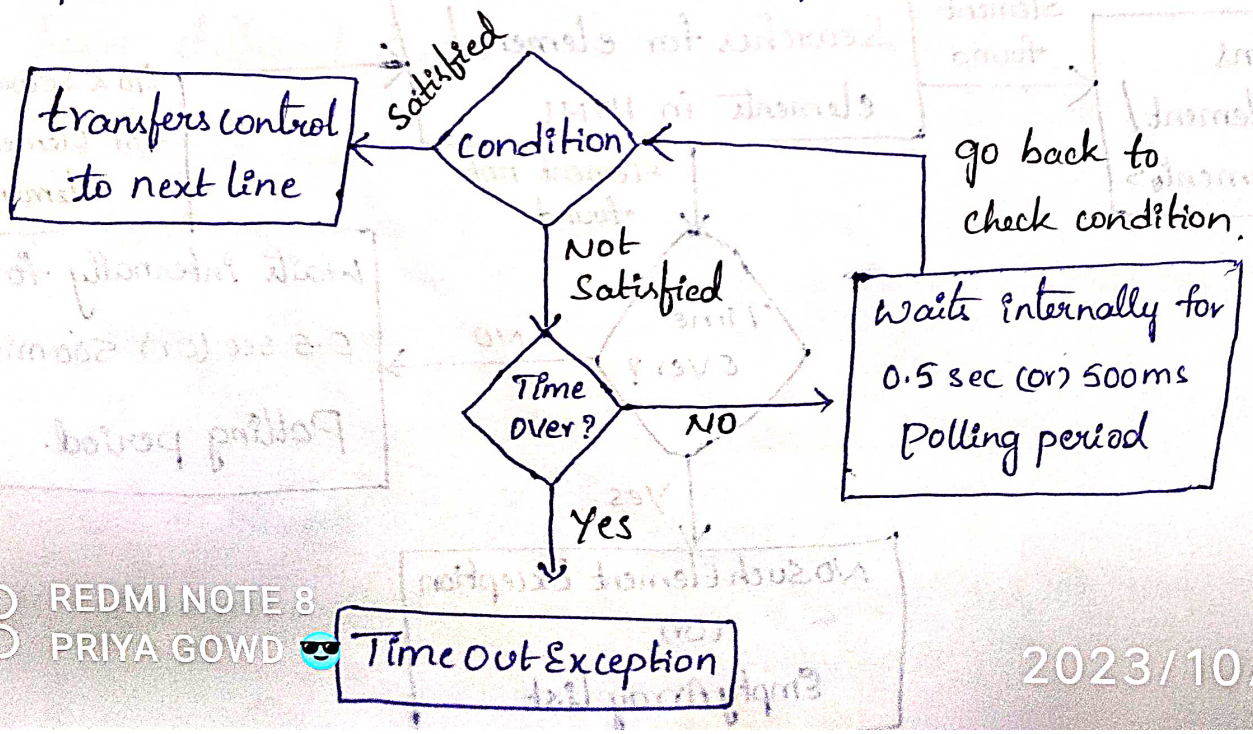


- Whenever implicitly wait is given, it searches for the element/elements in HTML., if the element/elements is found, it returns WebElement/ List<WebElement> respectively.
- If the Element is not found, it checks if the time is over or not
- If the time is over, it throws NO such element Exception (or) (Empty Array list) it returns Empty Array list.
- If the time is not over, it waits internally for 0.5 sec/500 milli sec. This waiting period is called as polling period.
- Once the polling period is done, it searches for the element/elements again.
- This process repeats until elements/element is found/time is over.

3. Explicitly Wait():

- It is Selenium Wait statement which synchronizes all the methods including findElement() & findElements():
- Unlike implicit wait, explicit wait should be specify whenever the synchronization is needed.

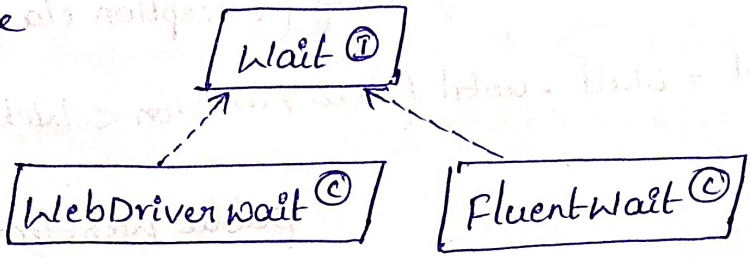
In Explicit wait, condition should be specified.



- Explicitly wait can be given using two classes

1. WebDriverWait ©
2. FluentWait ©

- Both WebDriverWait & FluentWait are implementing classes of Wait Interface



WebDriverWait :

Syntax:

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(time in seconds));

wait.until(ExpectedConditions.visibilityOf(element)   // returns WebElement
                             elementToBeClickable(element)
                             titleContains("<title of the page>"));   // returns Boolean
```

FluentWait:

- It is selenium wait statement and part of Explicit wait which is used to customize polling period and it also ignores any exception that might occur before the timeouts.

- Unlike webdriverwait, Fluent wait does not have predefined methods to build condition.

Syntax /(Generic method for Fluent Wait):

```
Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
    .withTimeout(Duration.ofSec(timeinsec))
    .pollingEvery(Duration.ofSec(timeinsec))
    .ignoring(Exception class);

WebElement element = wait.until(new Function<WebDriver, WebElement>()
{
    public WebElement apply(WebDriver driver)
    {
        WebElement e = driver.findElement(<locator>);
        if(e.isDisplayed())
        {
            return e;
        }
        else
        {
            return null;
        }
    }
});
```

get and Navigate Method differences.

get()

- It is used to navigate to an app, & waits until the page is loaded.
- It doesnot store Browsing history
- Return type is void.

Navigate()

- It is used to navigate to an app.
- It stores browsing history.
- Return type is Navigation.

Close()

- It is used to close the current tab/window & exits WebDriver

quit()

- It is used to close all the tabs (or) windows & kills the WebDriver process & exits.

WebDriverWait

- We cannot customize polling period
- We have predefined methods & to provide condition.

FluentWait

- We can customize polling period.
- We donot have predefined methods to provide condition, user should develop generic method to provide Condition.

Implicitly Wait

- It Synchronizes only findElement() & findElements() methods.
- It is global wait statement which is given only once in the beginning of the test script.
- Need not mention condition.
- It throws NOSUCHElement Exception/ returns Empty ArrayList if the element/elements is not found respectively.

Explicitly Wait

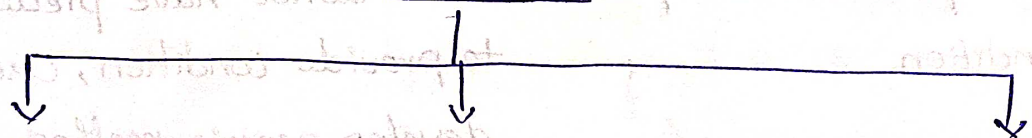
- It Synchronizes all the methods including findElement() & findElements()
- This wait should be specified every time whenever synchronization is needed.
- Need to mention condition that is mandatory.
- It throws TimeOut Exception.

Custom Wait:

It is user defined wait statement, developed using the concepts of Exception handling & loops.

```
int count = 0;
while (count < 20)
{
    try {
        WebElement element = driver.findElement (<locator>);
        break;
    } catch (Exception e)
        {
        count ++;
        Thread.sleep(1000);
    }
}
```

WebElement (I)



Actions
- Send keys ()
- Click ()
- Clear ()
- Submit ()

Getters
- get Text ()
- get Attribute ()
- get Tagname ()
- get Location ()
- get Size ()
- get Screenshot As ()
- get Rect ()
- get Css Value ()

Verification
- is Displayed ()
- is Enabled ()
- is Selected ()

→ The elements which appears on the web page is called Web Elements.

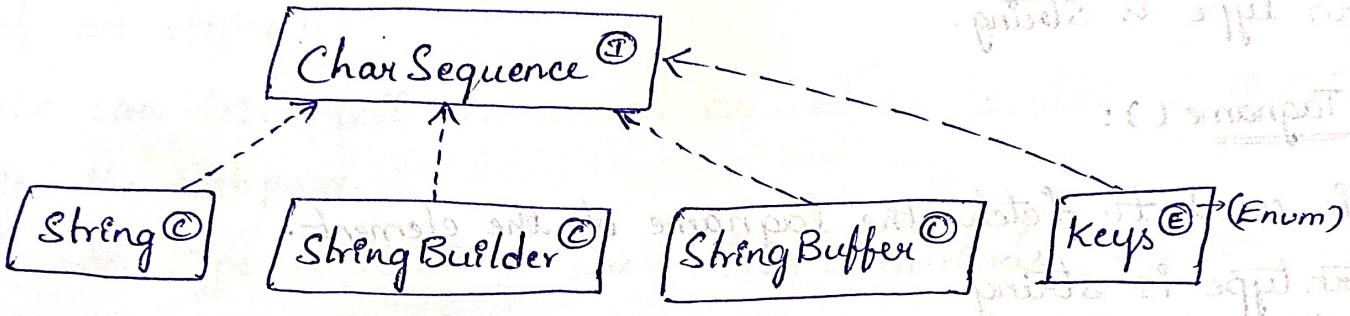
→ In order to handle web elements, we have an interface called WebElement interface.

→ This interface contains abstract methods categorised as follows: (see previous page)

Actions:

1. Sendkeys():

- It is used to pass data into the text field.
- It takes CharSequence Ⓘ type arguments.



- Returntype is void.

2. click():

- It is used to perform click action on an element.
- Return type is void.

3. Clear():

- It is used to clear the data in the text field.
- Return type is void.

4. Submit()

- It is similar to click() method but it is used only for the forms.
- The element should have type = "submit" attribute.
- Return type is void.

Getters:

1. getText():

- It is used to fetch the text on the element.
- Return type is string.

2. getAttribute():

- It is used to fetch the attribute value of the specified attribute name from the element node.
- It ~~acd~~ accepts attribute name in the string format as argument.
- Return type is string.

3. getTagname():

- It is used to fetch the tagname of the element.
- Return type is string.

4. getCssValue()

- It is used to fetch the specified Css property of an element.
  Css properties - color, background colour, ~~prop~~ font etc;
- It accepts the css property key in the String format as argument
- Return type is String.

5. getLocation()

- It is used to fetch the location of an element on the webpage.
- Return type is point class.
- The location is given in 'x' & 'y' coordinates using the methods getX(), getY()

6. getSize()

→ This method is used to get the dimensions of an element.

→ Return type is Dimension class.

→ to get the Height & width of an element, we have methods called getHeight(), getWidth() in the Dimension class. The returntype of these methods is int.

7. getRect()

→ This method is used to get the rectangular measurements of an element.

→ we can fetch both dimensions as well as location of the element on the webpage.

→ Return type is Rectangle class, has 4 methods : getX(), getY(), getHeight(), getWidth(),

→ These 4 methods returntype is int.

8. getScreenshotAs()

→ This method is used to get the Screenshot of an element.

→ We should use the external libraries named Apache commons io to fetch the Screenshot.

Steps to follow:

I. File image = element.getScreenshotAs(OutputType.FILE);

II. File destination = new File("<path.png>");

III. FileUtils.copyFile(image, dest);

Verification

~~isDis~~ isEnabled():

- This method is used to verify, if an element is enabled (or) not.
- Return type is boolean.

isDisplayed()

- This method is used to verify, if an element is displayed on the web page (or) not
- Return type is Boolean.

isSelected()

- This method is used to verify, if an element is selected or not.
- Return type is Boolean.

findElement():

- It is used to fetch a web element from the web page.
- It fetches the first matching element.
- It accepts locator as argument.
- Return type is WebElement.
- If the element is not found, it throws NoSuchElement exception.

findElements():

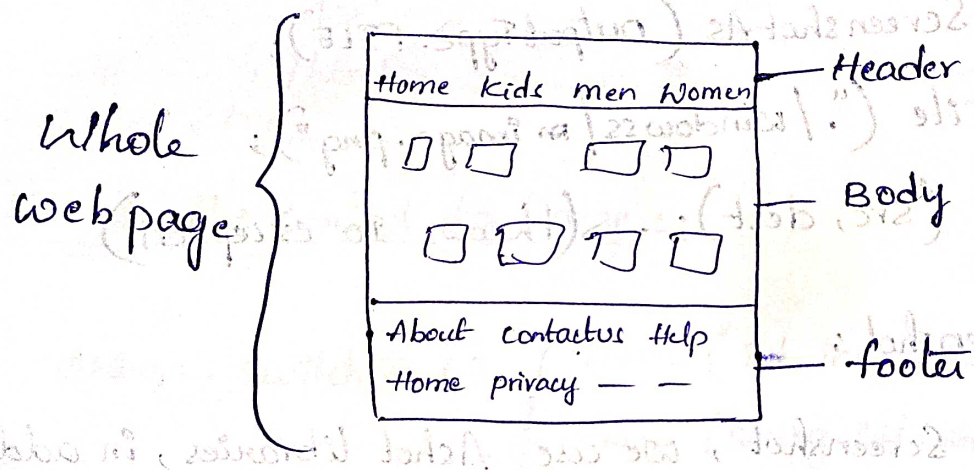
- It is used to fetch all the matching elements from the webpage.
- It accepts locator as argument.
- Return type is List<WebElement>.
- If the elements are not found, it returns EmptyArrayList.

Limiting driver Scope:

- The scope of driver is the complete Web page.
- It can be limited by fetching the path of Web page using findElement and using the element reference instead of driver

Example:

```
WebElement footer = driver.findElement(By.xpath("//div[@text()='footer']"));
footer.findElement(By.xpath("//a[text()='Home']")).click();
```



1, Screenshots :

* Window Screenshot :

- It can be taken with the help of TakesScreenshot interface and apache commons io libraries.

Steps to take Window Screenshot :

```
TakesScreenshot ts = (TakesScreenshot) driver;  → (Typecasting)
File src = ts.getScreenshotAs (OutputType.FILE);
File dest = new File ("./windowss/image.png");
FileUtils.copyFile (src, dest);  → (throws IO exception)
```

- Full Webpage Screenshot :

- For full webpage Screenshot, we use Ashot libraries, in addition to apache commons io libraries.

```
Screenshot screenshot = new Ashot().shootingStrategy(ShootingStrategies.
                                                    viewportPasting(1000))
                                                    .takeScreenshot(driver);
ImageIO.write(screenshot.getImage(), "PNG", new File("./windowss/
                                                    image.png"));
```

(variable name)

"PNG" → Type of image

new File → address to Create space for image.

2. Frames

- These are the webpages which are developed separately & kept inside the main webpage.
- Frames can be identified in 2 ways.
  1. When we right click on a frame, we can observe an option, 'View frame Source'
  2. When we inspect the frame element, the following code appears.

```
<iframe id=" " , name=" ">
  <html>
    :
  </html>
</iframe>
```

to switch the control to frame :

```
driver.switchTo().frame(int index);
                  frame(String id/name);
                  frame(WebElement frameElement);
```

to switch back the control to default webpage.

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

* SwitchTo() :

- It is one of the abstract methods of WebDriver interface which is used to switch the control to either window (or) frame (or) popup
- Return type is TargetLocator (SI).

(SI – Static Interface)

Actions©

Mouse Actions

- Mouse actions can be automated using Actions©.
- Actions© is present in "org.openqa.selenium.interactions"

Steps to perform Mouse actions:

I Actions a = new Actions(driver);

II
mouse hover → a.moveToElement(element).perform();
double click → a.doubleClick(element).perform();
right click → a.contextClick(element).perform();
drag and drop → a.dragAndDrop(element, target).perform();

Auto Suggestions:

This can be handled using findElements() method.

findElements():

It is one of the abstract methods in search context interface

- It is used to fetch list of all the matching elements from the webpage.
- It accepts locators as argument.
- Return type is List<WebElement>
- If the elements are not found it returns Empty Array list.

Static dropdowns:

→ The dropdowns in which the elements are fixed (or) does not changed are called static dropdowns.

→ These are identified by <select> tag.

```
<select>
    <option value=" "> text </option>
    .
    .
    .
</select>
```

→ 2 types: Single select & Multi select

Single select	Multi select
– The dropdown where we can select only one element at a time.	– The dropdown where we can select more than one element.
– Deselection of the element is not possible ↓ (unsupported operation Exception) occurs	– Deselection is possible.

→ Static dropdowns are handled using 'Select' class.

(org.openqa.selenium.support.ui) → package.

Steps to handle Static dropdowns:

I. Select s = new Select (element);

II. To select an element —
- s.selectByIndex (int index);
- s.selectByValue (String value);
- s.selectByVisibleText (String text);

→ Return Type is void

- To check if the dropdown is single select / Multi select

↓

`S.isMultiple();` → Return type is Boolean.

- To get all the options from dropdown – `S.getOptions();`

↓

Return type is List<WebElement>

- To get first selected option – `S.getFirstSelectedOption();`

↓

Return type is WebElement.

- To get all selected options – `S.getAllselectedOptions();`

↓

Return type is List<WebElement>.

- To deselect –

```
S.deselectByIndex(int index);
S.deselectByValue(String value);
S.deselectByVisibleText(String Text);
S.deselectAll();
```

→ Return Type is Void.

Javascript Executor ①

- This interface is used to execute JavaScript code in selenium, it is present in selenium webdriver class diagram.

Features / Advantages:

1. Handle scroll bar
   - Hard coding coordinates
   - Using element location
   - Using element reference
2. Navigate to an application
3. Refresh the web page
4. Fetch title & url of the webpage
5. Pass data into text field
6. Perform click operation on element
7. Handle shadow-root elements
8. Handle disabled elements.

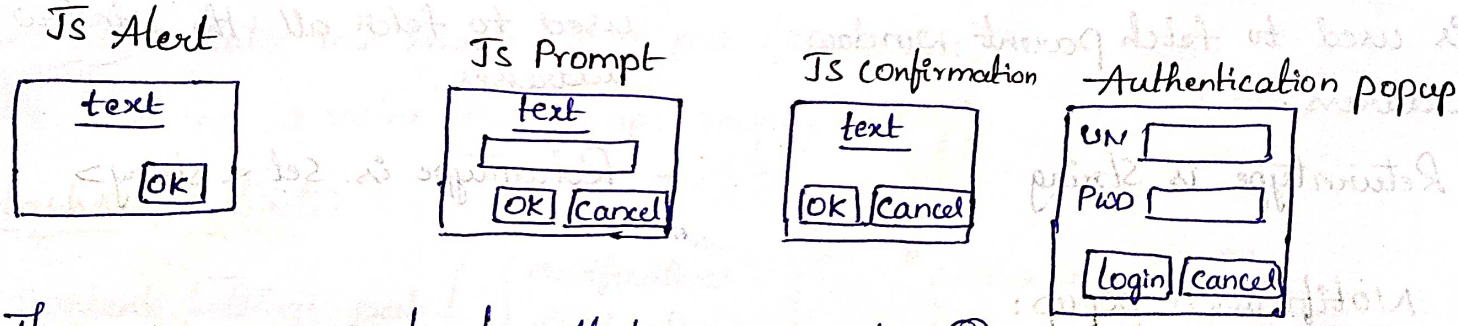
Popups

- Popups are the ~~sar~~ small pages which appears on the webpage that gives vital information to the user.
- We have diff. types of popups.
  1. Alert / Javascript popup
  2. Hidden Divison / Calendar popup
  3. Child Browser popup
  4. Notification popup
  5. File upload popup
  6. File download popup.

1. Alert popup:

- These popups are developed using java script code.
- These are not inspectable & not Movable

Types of in javascript popups:



- These popups can be handled using Alert (I) interface.

```
Alert (I) al = driver.switchTo().alert();

    Ok → al.accept();

    Cancel → al.dismiss();

To get text on popup → al.getText();

To pass data to popup → al.sendKeys();
```

Authentication popup / Login popup :

→ also called as Login popup.
→ It is not inspectable, not movable.
→ It can be handled by passing credentials like username & password in url.

Syntax:

```
driver.get("http://" + username + ":" + password + "<url>");
```

2. Hidden division / calendar popup :

- It is not movable but inspectable.
- This popup is handled using findElement() method.

3. Child Browser popup :

- It is handled using getWindowHandle() & getWindowHandles Methods.
- This is inspectable & movable.

getWindowHandle()

- It is webdriver Method which is used to fetch parent window address.
- Returntype is String

getWindowHandles()

- It is a webDriver method which is used to fetch all the window addresses.
- Returntype is Set<String>

4. Notification popup :

- These popups are browser specific & can be handled using browser specific classes.

- Chrome – ChromeOptions ©
- firefox – firefoxoptions ©
- Edge – EdgeOptions ©

- These popups are not inspectable & not movable.
- These popups are of 3 types. (66)

(1) Notifications
(2) Geo location
(3) Media-stream.

→ Steps to handle these popups.

1. Create an instance of browser specific class

ChromeOptions = new ChromeOptions();

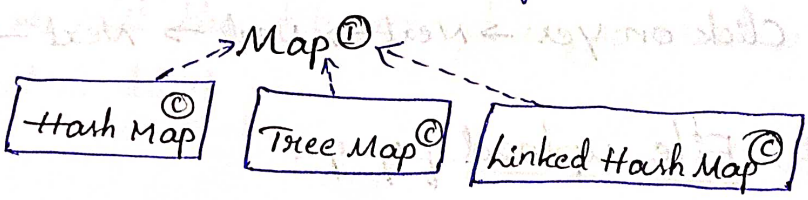
2. Using instance, we can call respective methods.

To disable Notification ⇒ Options.addArguments("--disable-notification");

To disable Geo-location ⇒ options.addArguments("--disable-geolocation");

Map (I)

Key-value pairs
Duplicate keys not accepted
only 1 null for key



- To store data into map: map.put(key,value);
- To retrieve a value from map: map.get(key);

HashMap (C) objects:

- Content settings.put("notifications / Geolocation / media-stream", "0 → default / 1 → Allow / 2 → Block");
- Profile.put("managed_default_content_settings", Content settings);
- Preferences.put("profile", profile);
- ChromeOptions (C) ⇐ Options.setExperimentalOption("prefs", preferences);

Steps to download AutoIt:

- Open the browser & type AutoIT download.
- click on first link
- Scroll down until software table.
- click on download AutoIT.

Steps to download scite Script Editor:-

- Same autoIt download page under software table.
- click on download Editor.
- Scroll down until current Versions & Click on SCITE4AutoIt3.Exe
- Go to downloads.
- Double click on SCITE4AutoIt3.Exe.
- Click on yes → Next → Next → Next → Install → Finish.

5, File upload popup:

→ It is movable but not inspectable using Selenium. Since, it is desktop application.

→ This popup can be handled in 3 ways.

1, Sendkeys() → type = "file"

2, Robot ©

3, AutoIt.

1, Sendkeys():

Wherever we have type = "file" attribute in the element node we can use sendkeys to upload file.

AutoIt Script steps:

Steps to launch SciTE Script Editor:-

- Type Scite Script Editor in search
- open the app.

Standalone application Scenario steps:

- Switch to the opened file popup window.
- Switch the control to filename – text bar.
- type file path.
- Click on open button.

Step 1: WinWait Active ("title")

This method is used to wait until the file upload popup window is active & switches the control to it.

Step 2: Sleep (2000)

In autoIT sleep() is the only wait statement.

Step 3: ControlFocus ("title", "text", "controlID")

- In order to inspect an element in standalone application.
  1. Search for 'autoIT v3 window info' in Computer.
  2. Drag & Drop the 'finder tool' to the element to be inspected.
- control focus method is used to switch the control over the element.
- Control ID: It is the combination of class & instance.

Step 4: Send ("Path_of_the_file_to_be_uploaded")

- Send method is used to type data into the element.

Step 5: Sleep (2000)

Step 6: Control click ("title", "text", "control ID")

Control click method is used to click on the Element.

- After writing the program, save it in a folder in desktop with au3 extension.
- To compile the program in Scite Script Editor, 'click on Tools' & Click on 'Compile' & click on 'Compile Script' button.
- We can see .exe file auto-saved to the folder on desk top.
- Now integrate the above code with selenium program.

RunTime.getRunTime().exec("path of .exe file");

Data Driven Testing

The process of driving the data into Test scripts from external files and performing Test Execution, is called Data driven testing.

Why Data driven Testing?

As per the rule of Automation, data should not be hardcoded because data modification and maintenance is tedeous (difficult) job when we want to run the test with different data. That's because we should get the data from external resources.

Advantages of Data driven Testing:

- Maintenance of Test data is easy.
- Modification of Test data is easy.
- Cross browser (or) platform testing is easy.
- Running Test scripts with different data is easy.
- Data sharing among team members is easy &
- Data Reusability can be achieved.

---

* Data is of 2 types: 1. common data
2. Test data

Common data:

The data which is common to all the Test scripts

Ex: Url, timeouts, browser, jdbc url

Test data:

The data which is Test Script Specific.

Ex: Naukri registration (all personal data - name, DOB, Qualification...)

Properties file:

It is a java feature file where we can store the data in the form of "key-value pairs."

- The data type of key & value should be always "String".
- properties file has '.properties' as extension.
- Different ways to store data in properties file.

```
1 browser = chrome
1 browser : chrome
1 browser chrome
```

Why properties file?

- Property p. file is light weight & faster to read data compared to any other file.
- java has it's own library to read data from properties file.
- we store common data in properties file.

—— × ——

* In properties file map interface concept is used.
* Whenever property. load (fis) is given, it internally creates Hashtable (It is the underlying data structure of HashMap©). and fetches all the data from this table.

Drawbacks of properties file:

- Only single data is fetched at a time.
- We should remember all the keys to fetch the values.
- Data is not stored in organized manner.

Excel file:

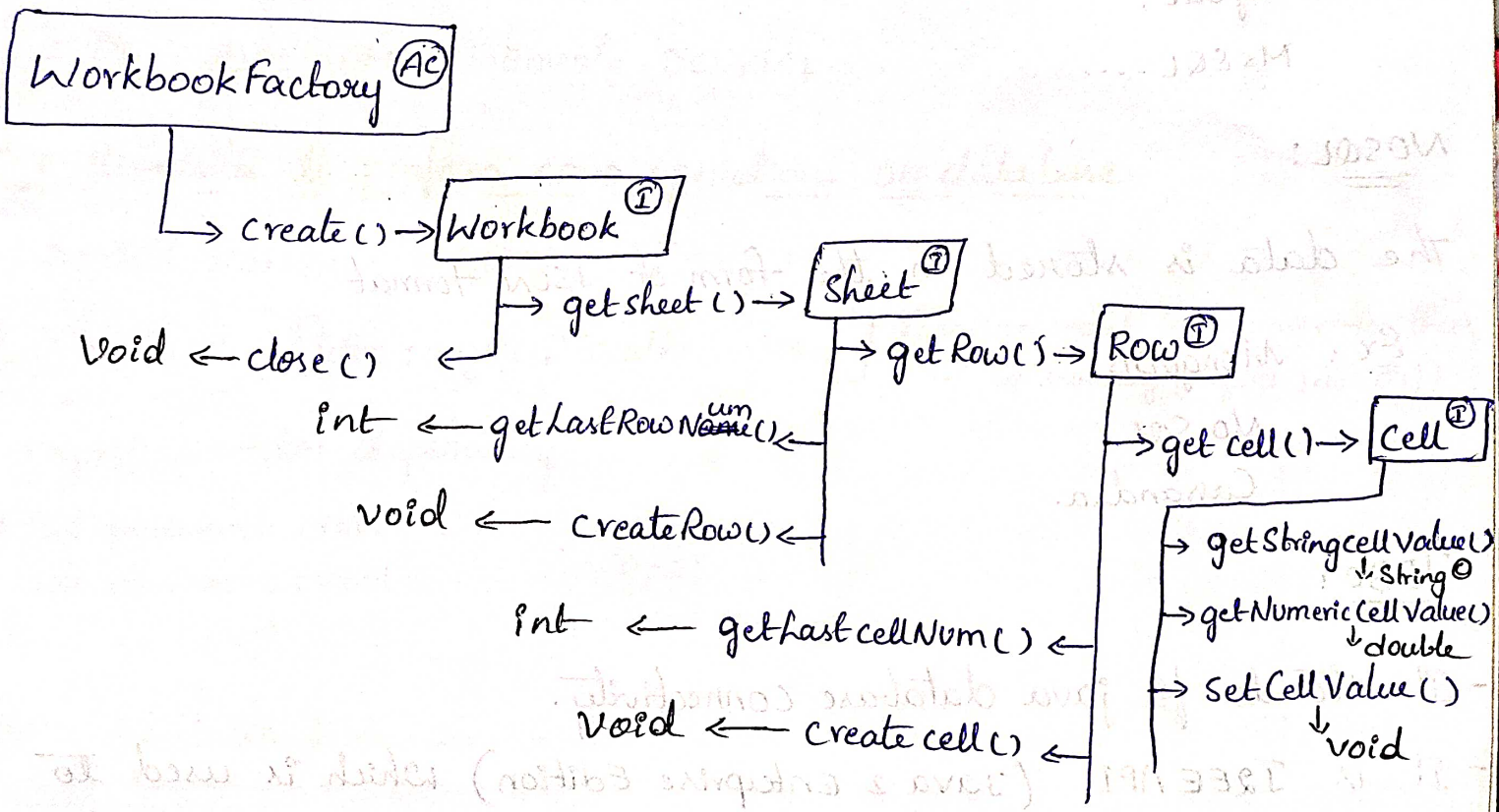
→ The data is stored in tabular form in excel.

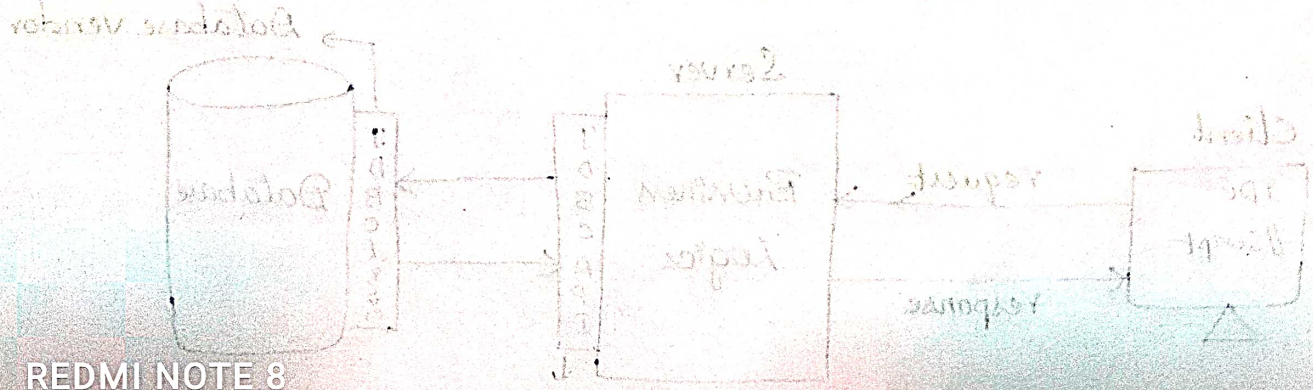
→ So, we use Apache poi external libraries to perform operations on Excel.

→ Apache poi libraries are used to read data from any microsoft documents. (doc, docx, pptx, xlsx, txt, pdf .....)

→ Test data is stored in excel file.

Apache poi class diagram:





Data Base:

It is medium or a place where we can store the data in a systematic manner.

- we have 2 types of databases
  1. SQL
  2. NOSQL

SQL:

The data is stored in the form of tables.

Ex: Oracle 10g,
MySQL,
MsSQL....

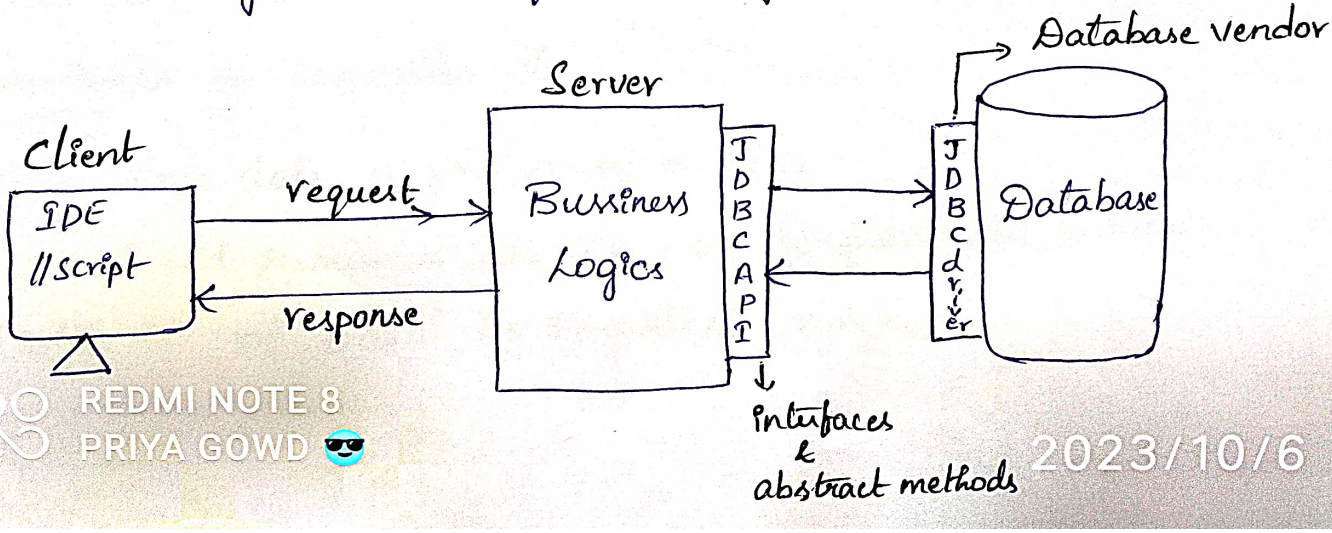
NOSQL:

The data is stored in the form of JSON format

Ex: MongoDB
NoSQL.
Cassandra.

JDBC:

- It stands for java database connectivity.
- It is J2EE API (Java 2 Enterprise Edition) which is used to connect to any database from the java code.



- JDBC is required for both Developers and Automation Test Engineers

Developers use JDBC to communicate with Database.

Automation Testers use to automate the preconditions

Types of SQL Queries:-

Select Query: Any Query which has select clause

Ex: All DQL statements

Non select Query: Any Query which do not have Select clause.

Ex: DDL, DML statements, DCL, TCL

Pre requisites to perform CRUD operations on database

1. Install database (MySQL)
2. JDBC url [jdbc:mysql://localhost:3306/] (we can search in brooser for this url by searching (mysql jdbc url))
3. mysql connector dependency
4. DB password : root
   DB username : root

POM
[page object Model]

Object Repository:

It is the collection of elements, locators and bussiness libraries in one place.

Why Object Repository?

As per the rule of Automation, we should not hardcode elements within the Test scripts because modification and maintenance of elements becomes difficult, instead we should get elements from object Repository

What is POM?

POM is a java designed pattern preffered by Google to develop Object Repository.

Why POM?

It is a well organized structured designed pattern where we can maintain all the web elements page wise and also it follows Encapsulation concept hence elements can be stored in secured manner.

Advantages of POM:

- Maintenance & modification of elements is easy & fast bec
- Test script development is faster due to bussiness libraries.
- Test script code can be optimised.
- Increased readability of code.
- Reusability of elements & bussiness libraries
- Handles StaleElement Refrence Exception.
- Debugging is easy.
- Supports Auto Healing feature.

Rules to develop POM class:

I. Declaration → @FindBy | @FindAll | @FindBys

```
@FindBy (LocatorName = "Locator value")
private WebElement / List<WebElement> elementRef;
```

Fetches element and stores in element Ref.

Works like (OR) operator

```
@FindAll (@FindBy (LN1="LV1"), @FindBy (LN2="LN2"), .....)
private WebElement / List<WebElement> elementRef;
```

Works like (AND) operator.

```
@FindBys (@FindBy (LN1="LV1"), @FindBy (LN2="LN2"), .....)
private WebElement / List<WebElement> elementRef;
```

II. Initialization

```
public ClassName (WebDriver driver)
{
  PageFactory.initElements (driver, this);
}
```

III. Utilization.

We provide getters to access the elements.

Annotation @FindBy

- It is used to identify one element with one locator.

Annotation @FindAll

- It is used to identify an element with multiple locators.
- It works like OR operator.
- Autohealing feature is achieved through @FindAll

Auto healing:

- During execution, if one locator fails to identify the element, it will retry to identify the same element using another locator.
This technique is called as Auto healing

@FindBys:

- It is used to identify an element using multiple locators.
- It works like 'and' operator.

What is Pagefactory?

- It is extended designed pattern of Pom which is used to create an object to Pom classes and at the time of object creation, it executes all @FindBys and @FindAll, then initializes all the elements.

What is StaleElementReferenceException?

It is Selenium Exception which occurs when we try to access an element using old (or) Expired reference.

It is unchecked Exception.

How Pom Handles StaleElementReference Exception?

pom will identify the web elements with annotations like ~~annotat~~ @FindBy, @FindAll, @FindBys instead of FindElement() (or) FindElements() methods.

- All these WebElements will be initialized to drivers current reference with pagefactory.initElements() method.

What is Encapsulation, How POM achieves Encapsulation ?

Binding the state & behaviours of an object in a class and limiting the access to the variables is called Encapsulation.

- In POM, all the WebElements are declared using private access specifier and we provide getters and and access the elements through getters. Hence, this makes WebElements secured and no unnecessary changes in element locators are entertained.

TestNG → testing framework which can be used for unit, integration, end to end & functional testing.
(Test Next Generation)

It is (Unit) Testing framework tool used by both Developers and Test Engineers. (Only for java programming language)

Jyothsna priya Mandapaka

Developers Usage:

- use for White Box Testing.

Test Engineer Usage:

- ~~use for~~
- used to develop framework for test suit execution.
- TestNG is developed as a separate plugin for IDEs.
- It is the combination of JUNIT (Java) & NUNIT (.Net) with advanced features.

JUNIT: It is unit Testing framework tool for java.

NUNIT: It is unit Testing framework tool for .Net.

- TestNG headless tool. (It doesn't have User Interface)
- IDEs provide U.I for TestNG.

Features in TestNG:

- prioritising Test cases
- Re running the Test scripts — with same Data
  — with different Data.
- Disabling the Test Scripts.
- Creating dependency between the methods.
- controlled flow of execution.
- Batch execution.
- Group execution
- parallel execution
- Automatic generation of reports.
- Assertions.
- Listeners to monitor test suite execution
- Re running failed Test scripts.

Steps to convert TestNG java class to xml file:

- Right click on the java class file.
- Go to TestNG → click on convert to TestNG
- Modify the xml file name & click on finish.

Automatic generation of Reports:

- TestNG has a special feature to generate html reports automatically.

Steps to view html Reports:

- Convert java class file to xml file.
- Execute xml file
- Refresh the project
- expand 'Test output' folder
- Double click & open Emailable - report.html.

Prioritizing the Test cases:

We can prioritize the Test cases by giving priority to each & every Test Method.

Usage:

```
@Test (Priority = int value)
  public void method name ()
  {
    // test script
  }
```

- priority follows Integer number line order of execution.
- Default priority is 'Zero'
- If the test scripts are having same priority / no priority given then it considers ascii values of method names of execution.

Rerunning the Test scripts with same data:

- It can be done using Invocation count (is a paramater/attribute which tells the test method, how many times to get executed)

Usage:

```
@Test (invocationCount = int)
Public void MethodName ()
{
  // Test script
}
```

- Default invocation count is '1'.
- If the invocation count is '0' (or) Negative number, it will consider the test script in the execution queue but the script will not be executed.

Data providers:-

- data provider is an annotation in testNG which is used to run the same test script with multiple times with different sets of data.
- The data in dataprovider is stored in two dimensional object array.

	0	1	2
0	From (0,0)	to (0,1)	Departure date (0,2)
1	Hyd (1,0)	Blr (1,1)	8th Aug (1,2)
2	Hyd (2,0)	Chennai (2,1)	9th Aug (2,2)
3	Hyd (3,0)	Kerala (3,1)	10th Aug (3,2)
		Delhi	11th Aug

- Return type of dataprovider method is always two dimensional (2D) Object Array.
- @DataProvider to be given in the same class where (annotation test) is present (@Test)
- Test class can have multiple @DataProviders but @Test can read only one dataprovider at a time.

Ex:-

```
@DataProvider
public Object [][] data()
{
    Object [][] obj = new Object [2][3];
                      // 2 -> No. of sets of data
                      // 3 -> No. of arguments in each set
    obj [0][0] = "Hyd";
    obj [0][1] = "Blr";
    obj [0][2] = "8th Aug";
    obj [1][0] = "Hyd";
    obj [1][1] = "chennai";
    obj [1][2] = "7th Aug";
    return obj;
}
```

```
@Test (dataprovider = "data")
public void bookTicket (String src, String dest, String date)
{
  Sopln (src + "\t" + dest + "\t" + date);
}
```

Disabling the Test Scripts:

- Test script can be disabled using the 'enabled = false'.
- By default enabled is 'true'.
- When we give enabled = false, it will not consider the Test script in execution queue and it will not execute.

Usage:

```
@Test (enabled = false)
public void test1()
{
  // test script
}
```

Creating dependency b/w the methods :-

It can be done using dependsOnMethods

Usage:

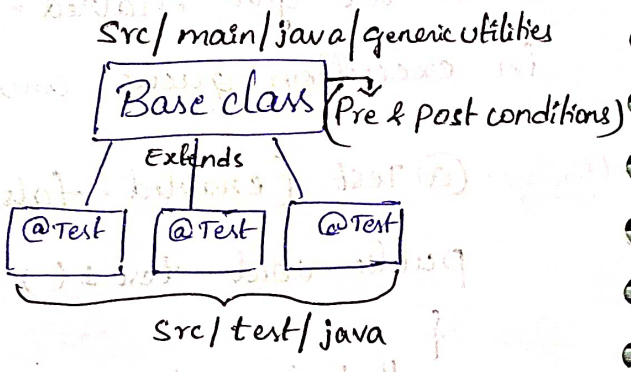
```
@Test
public void test1()
{
  // Test script
}
@Test (dependsOnMethods = {"test1", "test3", ....}) (Multiple methods we can pass)
{ Public void test2()
  {
    // test script
  }
```

alwaysRun = true:

If we wanted a Test script to be executed irrespective of the status of Execution of dependant methods, we provide alwaysRun = true.

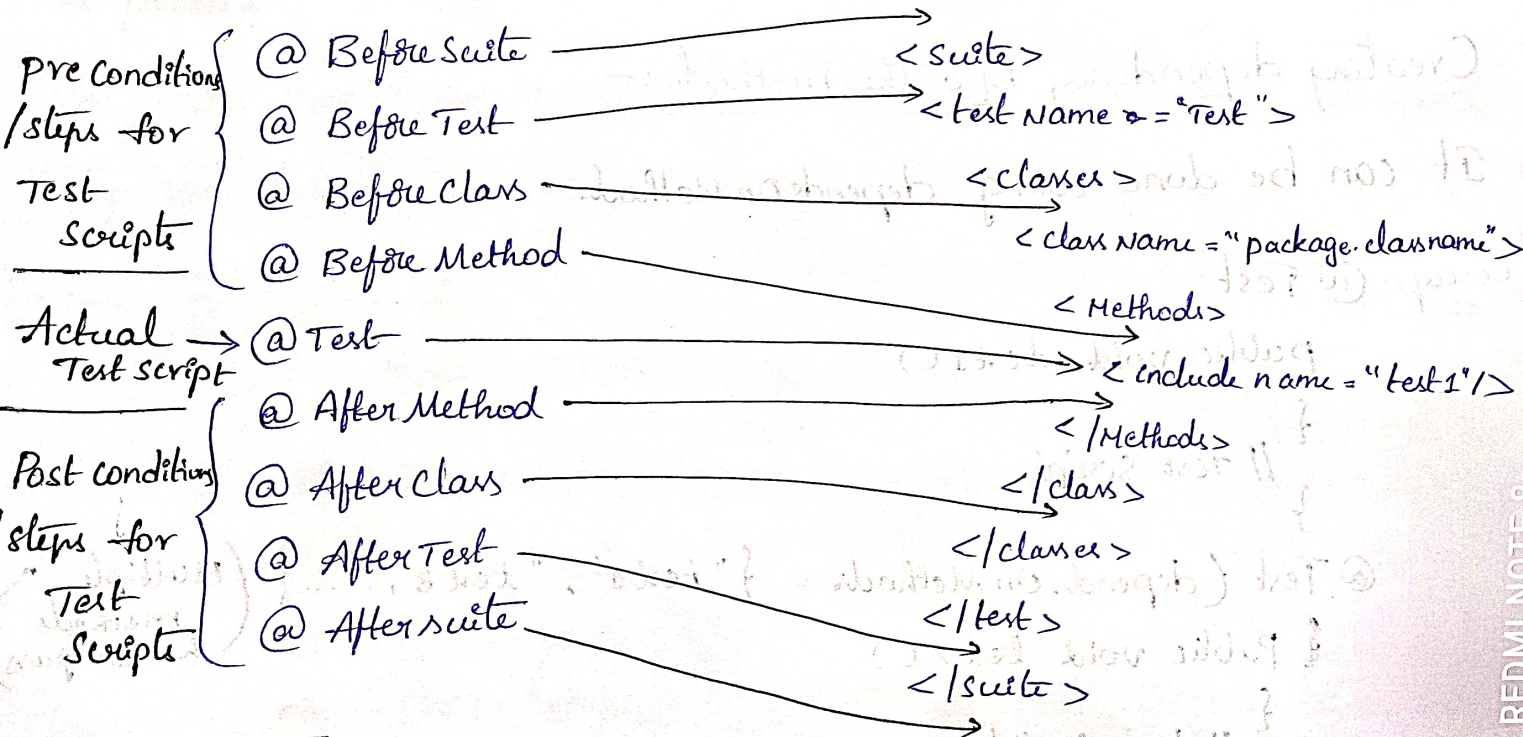
Usage:

```
@Test
public void test1()
{
  // test script
}
@Test (dependsOnMethods = {"test1".....}, alwaysRun = true}
Public void test2()
{
// test script
}
```



Controlled flow of Execution: (Annotations):

TestNG has a special feature called Annotations which facilitates Controlled flow of execution.



Base class → where we store pre & post conditions.

@ BeforeSuite:

- It is executed before the <suite> in xml file.
- It is executed only once per execution of xml. file.
- It is used for establishing database connection.

@ AfterSuite:

- It is executed after the </suite> in xml file
- It executes only once per execution of xml file
- It is used for closing database connections.

@ BeforeTest:

- It is executed before <Test> in xml file
- The no. of times it executes depends on no. of <test>
- It is used for parallel executions.

@ AfterTest:

- It is executed after </Test> in xml file.
- The no. of times it executes depends on no. of </test>
- It is used for parallel executions.

@ BeforeClass:

- It executes before the <class> in xml file / before every test class
- The no. of times it executes depends on no. of <class> & (or) no. of test classes.
- It is used for Launching the browser.

@ Afterclass:

- It executes after the <class> / before every test class in xml file.
- The no. of times it executes depends on no. of <class> / test classes.
- It is used for closing the browser.

@ Before Method:

- It executes before every @test
- The no. of times it executes depends on no. of @ test Methods.
- It is used for logging into the application.

@ After Method:

- It executes after every @ test.
- The no. of times it executes depends on no. of @test Methods.
- It is used for logging out the application.

@Test:

- It is the actual Test script.
- One @Test is equal to one Test script.
- A class can have more than one @Test Methods.
- It acts like main method in java for the JVM to start execution.
- Generally, in Real Time, a class will contain 10 to 15 @Test Methods.

Additional Annotations:

@ Parameters
@ DataProvider
@ Listeners.

→

Flow of Execution:

BS
BT
BC
BM
@Test
AM
AC
AT
AS

2 @Test

BS
BT
BC
BM
@Test (1)
AM
BM
@Test (1)
AM
AC
AT
AS

2 class

BS
BT
BC
BM
@Test
AM
AC
class (1)
BC
BM
@Test
AM
AC
class (2)
AT
AS

2 Tests

BS
BT
BC
BM
@Test
AM
AC
AT
Test (1)
BT
BC
BM
@Test
AM
AC
AT
&
Test (2)
AS

parallel Execution

Batch Execution:

Executing all the test cases sequentially is called Batch Execution.

Steps to conduct Batch Execution:

- Convert all the Test classes into single xml file.
- run the xml file.

Group Execution:

Grouping similar kind of test cases and executing specific group at a time is called group execution.

Steps to perform Group Execution:

1. Specify the group for each & every annotation.

```
@Test (groups = "Smoke")
public void tc1 ()
{
}
```

→

```
@Test (groups = {"smoke", "regression"})
public void tc2()
{
}
```

2. Convert test classes into xml file.

3. include / exclude the groups in Test Execution as follows.

```
<Suite name = "suite">
<groups>
<run>
<include name = "regression"/>
<exclude name = "Smoke"/>
</run>
</groups>
<test name = "test">
<classes>
<class name = "  " />
:
</classes>
</test>
</suite>
```

Parallel Execution:

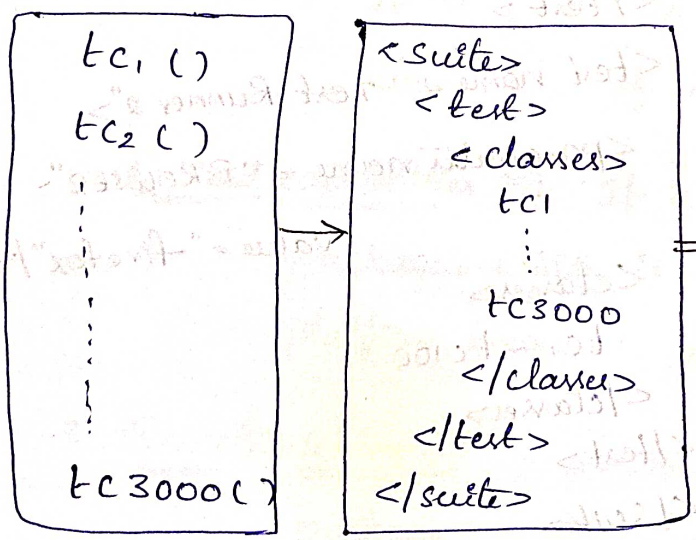
Executing the test classes (or) the test methods simultaneously by creating multiple threads is called as parallel execution.

Types in parallel Execution:

1. Distributed parallel
2. Cross Browser Parallel. } → Selenium Grid.
3. Cross platform

Distributed parallel Execution:

- Executing the test cases simultaneously by distributing them to different test tags (or) test runners is called distributed parallel execution.

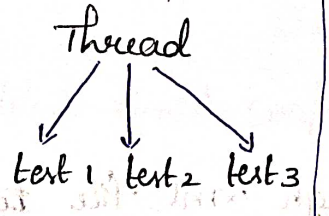



```
<suite name="suite" parallel="tests"
                    thread-count="3">
<test name="Test1">
<classes>
    tc1 ... tc1000
</classes>
</test>
<test name="Test2">
<classes>
    tc 1001 – tc2000
</classes>
</test>
<test name="Tests">
<classes>
  tc2001 – tc3000
</classes>
</test>
</suite>
```

3000 tcs

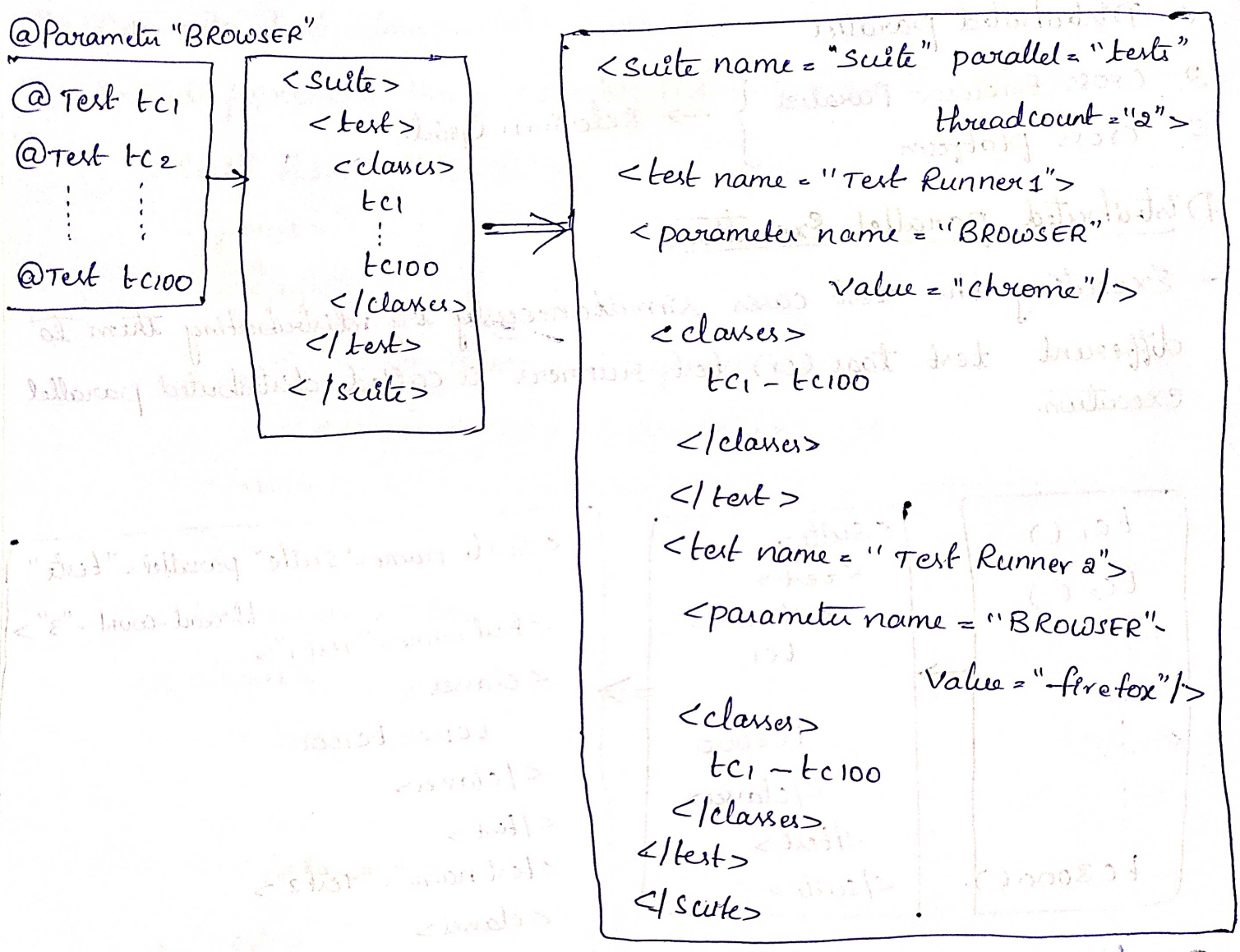
1000 tc → 8 hrs

3000 tc – 24 hrs



Cross Browser parallel Execution:

→ Executing same set of test cases on different browsers simultaneou
-sly is called cross Browser parallel execution (or) Browser Compatability test.



xml parameterization:

passing the parameters through xml file to the test script and performing test execution is called xml parameterization.

Selenium Grid:

- It is the collection of libraries, Selenium Rc, Selenium WebDriver, and Selenium Grid server.
- It is an open source available selenium community which is used to perform Remote Test Executions.
- Remote devices means Machines on same network, mobile devices, cloud Machines.
- Selenium Grid is used to perform - Remote executions
  - Cross browser parallel
  - Cross platform parallel.
- In order to perform Remote executions, we make use of 3 classes
  1. URL©: It is a class in java.net package which is used to store urls. of Remote devices.
  2. Desired capabilities©: It is a class in org.openqa.selenium.remote package, which is used to set/change the capabilities like platform, browser, etc... of WebDriver.
  3. RemoteWebDriver©: It is a class in org.openqa.selenium.remote package.

- In order to establish Grid configuration, we should setup a 'Hub' and 'Nodes'.

Hub: It acts like Master and receives (request) Command from selenium server. and Bypasses it to the nodes.

Node: It acts like slave and receives command from Hub and executes the command. Nodes should be registered to Hub.

Pre-Requisites for Selenium Grid:

- JDK should be installed.
- IDE
- Selenium standalone server.
- Driver executors
-

To start hub:

- Open command prompt
- java -jar <Selenium standalone-server jar path> -role hub -port <port no>

To start node:

java -Dwebdriver.chrome.driver = <path of chromedriver.exe> -jar <standalone jar path> -role node -hub <hub url> -port <port no>

To view Selenium grid console:

1, Copy the url to which nodes should be registered

Ex: http://192.168.137.1/grid/register/

2, Open new tab in the browser and paste the url.

3, Replace 'register' with 'Console' and click enter.

Cross platform parallel:

1. Selenium standalone server jar.
2. driver executable files.
3. on same network.

Note: disable vmware, firewall and any antivirus software.

Assertions

* TestNG has a feature called Assertions, in order to provide validations to the Test scripts.

- There are 2 types of Assertions.
    1. Hard Assert
    2. Soft Assert

Hard Assert:

- It is given using Assert©.
- Assert© is present in ~~org.testng.~~ org.testng.asserts package.

Work flow

```
@Test
public void test1()
{
    ↓
    ⊗ ——→ Assertion Error
    } skip
}
```

- Whenever HardAssert is given, if the validation fails in the middle of the method, it throws Assertion error immediately and skips the remaining statements in the method and transfers the control to the next method.
- Assert© has all static Methods.

* assertEquals (actual, expected);
* assertNotEquals (actual, expected);
* assertTrue (condition);
* assertFalse (condition);
* assertNull (value);
* assertNotNull (value);
* fail ();

Soft Assert:

- It can be given using SoftAssert ©.
- SoftAssert © is present in org.testng.asserts package.

Workflow

```
@Test
public void demo()
{
    ↓
    ⊗
    |
    assertAll(); ⟶ AssertionError
}
```

- Whenever SoftAssert is given, if the validation fails, it will continue with the execution of remaining statements in the method and throws AssertionError at the end of the method.
- SoftAssert has all non-static methods.
  * assertEquals (actual, expected);
  * assertNotEquals (actual, expected);
  * assertTrue (condition);
  * assertFalse (condition);
  * assertNull (value);
  * assertNotNull (value);
  * fail ();
  * assertAll();

- assertAll() is the mandatory method is SoftAssert which should be given at the end of the block.

Listeners:

- Listeners are the advanced feature in TestNG which monitors actual test execution and captures Run time events like pass, fail, (or) skipped status of test scripts and performs necessary actions provided by the user.

Interfaces in TestNG:

ITestListener (I)

ITestResult (I)

IRetryAnalyzer (I).

ITestListener (I)

- It is an interface in TestNG which monitors Test Execution and performs the necessary actions given by the user, it has several abstract methods.

onStart()

onTestStart()

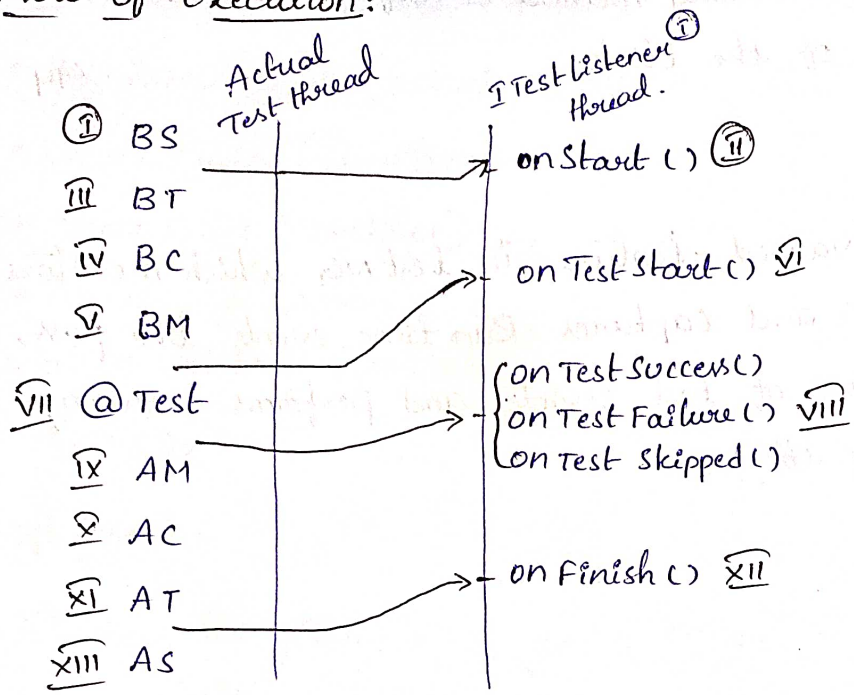
onTestSuccess()

onTestFailure()

onTestSkipped()

onFinish()

Flow of Execution:



- Generally, we configure reporting in ITestListeners

ITestResult (I):

- It is used to capture the result of Test script
- It is used as an argument but most of the methods in ITestListener (I)

IRetryAnalyzer (I):

- It is functional interface has one Abstract Method which is used to Re-run the failed testscripts automatically.
- Sometimes failures can be due to synchronization issues, network issues (or) server issues., if we Rerun the testscript for 3 (or) 4 times it might pass for these kinds of testscripts, we provide IRetryAnalyzer implementation.

Reports:

- TestNG generates automatic html reports when xml file is executed.
- It is only very basic reporting which provides log level Analysis. (Test script level)
- In these reports we cannot attach screenshots and there is no graphical representation & it is not customisable hence we go for "Extent Reports".

Extent Reports:

- It is a popular reporting format used in several companies as it gives graphical representation of high level report.
- Extent Reports can be supports used only with java & .net languages.
- Extent Reports can be configure in listener implementation (or) Baseclass of generic utilities package. (Most preferred one is Listener Implementation)

To add plugin's for TestNG:

Surefire plugin for TestNG.
↓
1st link
↓ (Scroll)
Using xml file
↓
Copy → paste in pom.xml → Inside build tag just before dependencies tag

Apache Maven download:-

1st link → click on Binary zip archieve link.

→

After downloading:

- Go to downloads & extract apache Maven Zipfile.
- Open extracted folder - copy the folder apache Maven 3.9.4
- Go to 'c' drive ; program files & paste the folder.
- Open Apache Maven - bin (open) - mover cursor to text field and copy the path.
- Right click on this PC - properties - click on advanced system settings Click on Environment Variables - under system variables, click on New → Give variable name as Maven Home → Paste the path in variable & value & remove /bin at the end. → Inside user variable double click on Path → Click on New → Paste the path → Click 'ok' → 'ok' → 'ok'

To check:

Open command prompt - type: mvn -version.

Jenkins Download:

- Go to download → Double click on jenkins file → click on Next → Next → Select Run service as Local system → Next → click Test port → Next → Check Java Version is JDK-17 → Next → Next → Next → Install → Finish.

Settings of the plugins in Jenkins:

Click on Manage jenkins → under system configuration → click on plugins

↓

click on Available plugins ← Type 'Maven Integration' in Text field ① ← 1st checkbox ← Click on install without restart

↓

Go back to available plugins → Github integration ② → Select check box → install without Restart

↓

Again Go back to Available plugins ← Search for Html publisher ③ ← Select checkbox ← Install without Restart

↓

Go back to available plugin → Search Build pipeline ④ plugins. → Select Checkbox → Install without Restart.

Steps for settings of the paths: JDK, Maven, Git

- Open a New tab → Git downloads → Enter → 1st link → windows

↓

(Standalone Install) 64-bit Git for Windows setup.

- Manage jenkins → System configuration → tools → under JDK → Add JDK

↓

Deselect Install automatically Check box ← Give name as Java-Home (files) ← Go to file ← C-program ← Java

↓

JDK-17 → copy the path in text field

↓

paste it under Java-Home.

- Under Git installations → give the name as Git → Copy the path → Go to file → C-program files → Git → bin → Copy the path → Paste it in before git.exe → type a '\' after Git.

- Scroll till maven → Click on Add Maven → Default install Automatically → Give the name as Maven-Home → Copy the path file → C-program files → Apache Maven → Copy the path → Paste in Maven_Home → click on Apply & Save.

Extent Reports:

- Extent Reports has 3 major classes
  1. Extent Spark Reporter
  2. Extent Reports
  3. Extent Test.

Extent Spark Reporter:

It is used to set the basic configuration of the report.

Extent Reports:

It is a main class which is used to set system information & browser information to the Reports.

Extent Test:

It is used to perform test script level Analysis of the Report.

Steps to generate Github token:-

- Login to Github account.
- click on profile Icon at Top right corner.
- click on settings.
- click on Developer settings in side Menu.
- click on personal Access Tokens.
- Select Tokens (classic)
- Select Generate New Token.
- click on Generate New Token (classic)
- Enter the Token name in Note.
- Set Expiration period (optional)
- click on Repo checkbox.
- Scroll down and click on Generate Token.
- Copy the Token & save it in Notepad or any other Secured place.

Steps to create New Repository in Github:

- click on + button on the Top.
- click on New Repository.
- provide Repository name
- Scroll down and click on Create Repository.

Steps to commit the project to local .git folder

- Open Eclipse, right click on the project → Go to 'Team' → Share project → Select use (or) create Repository in parent folder checkbox. → Click on the project ← Click on Create Repository ← Finish

- Right click on the project again, Go to Team
  ↓
  Click on COMMIT
  ↓
  In git staging, click on ++ (double plus) icon
  ↓
  All the files will be shifted from unstaged changes to staged changes.
  ↓
  Enter commit message (~~any name~~ Framework first commit)
  ↓
  Click on Commit

Steps to push project to Global Repository :-

- click on Push Head
- Go to github Repository, copy the git url.
- paste it in URI section., Host & Repository path are Autofilled.
- provide github user name and git token as password.
- click on Preview.
- click on preview again.
- click on push.

- Provide username & github token again & click on login.
- close
- Go to github Repository & refresh the web page.

Steps to clone and import the project from github to Eclipse:

- Go to Eclipse, click of File → click on Import → Expand git folder
↓
click on projects from git (with smart import)
↓
Click on 'Next' → Clone URI → Next
- Go to github → open the Repository → click on 'code' → copy the URL and paste it in URI section (Eclipse)
↓
Provide username and github token → Next → Next → Next → Finish

Capabilities :-

- It is an interface in selenium, present in org.openqa.selenium package
- It describes the properties of browser in the form of key value pairs.
- It has methods like -
  - getBrowserName()
  - getBrowserVersion()
  - getPlatformName()
  - getCapability(String capabilityName)

  } String

Git

- Git is a version control tool which is used to manage and track the changes made to code and other files.
- It has two softwares 1. Github
  2. Git client

Github:

— It is distributed decentralized cloud based Repository where we can maintain source code (or) Automation frame work (or) CRS doct. (or) software versions in one place.

- It can be accessed using URL.

Git client:

— It is the software which should be installed in client Machine.

- It is used to communicate to Github.

Eg: Egit, Git(Bash)

Egit:

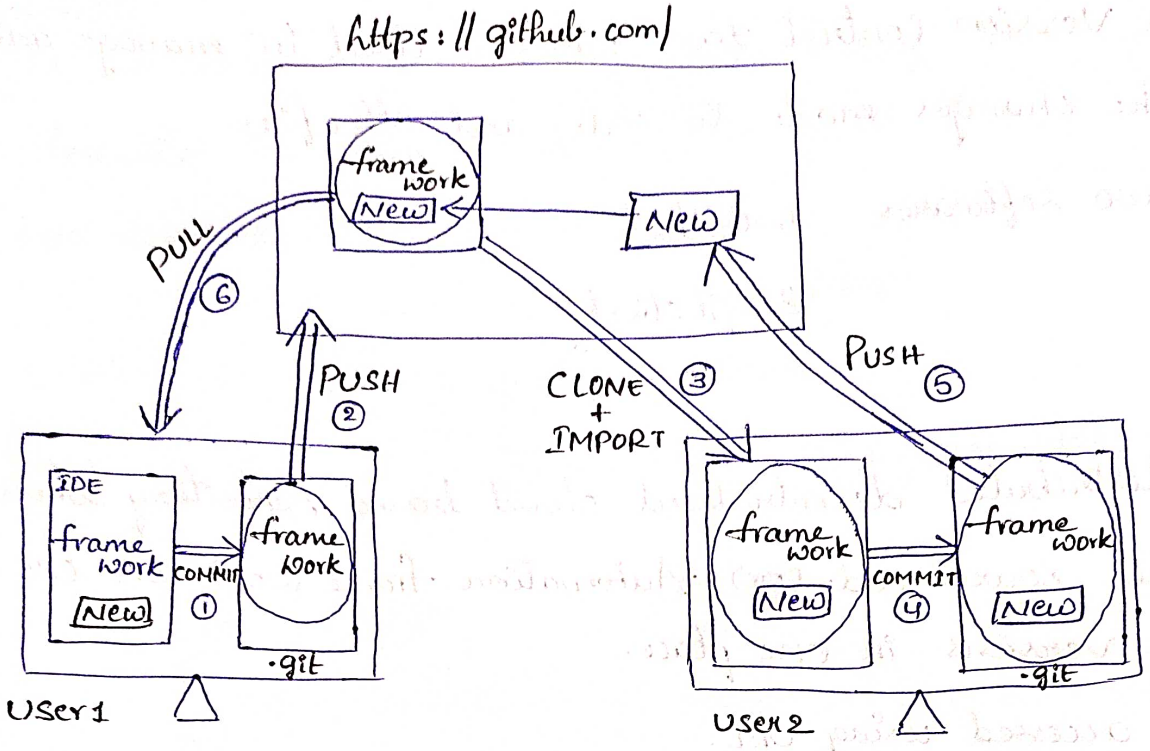
It is inbuilt in IDEs, which allows us to perform several operations like commit, push, pull, clone, import

Git (Bash):

- This software should be explicitly installed in our local system.
- It is used to perform all Github related operations through command line.

→

Git Architecture:



Why git is decentralized repository?

In git, before pushing any code to github, we have to commit the code to local repository (.git) first and make sure the code is working properly, only then we push the code to github.

- git is used by developers, Automation testers, devops and Manual Test Engineers.

Developers usage:

used to maintain source code of the Application in one place.

Manual Test Engineer usage:

used to maintain entire CRS (or) Use case document of the Application.

Automation Testers usage:

used to maintain Automation framework in one place.

Devops usage:

Used to maintain multiple Application build version in one place.

Advantages of github:-

- Since, it is a cloud based Repository, no need to have maintenance fee to maintain software (or) Hardware.
- We can access the repository through internet from any place.
- Initial investment is not required.
- Scale up (or) Scale down is easy.
- File sharing among team members is easier.
- Contributers can access our repository from anywhere.
- It provides history of changes made by the users.
- It provides platform to review the code changes.
- It also handles conflicts.
- jenkins always gets the latest frame work from github for Batch execution.

Steps to pull the code:

Right click on the project → Team → click on pull.. → finish → close

What is git conflict:-

When two (or) more engineers try to make changes in the same repository simultaneously, the first user will not face any issue when he pushes the code. but when second user tries to push the code github will throw conflict and it will give Rejection message.

It is because github thinks the second user is not aware of changes made by the first user.

Solution for conflict:

- pull the code first before pushing.
- Remove all unwanted changes.
- Analyse the code pushed by others.
- push the code.

Basic git commands:-

commit:

It will copy the framework from working directory to local repository (.git folder).

push:

It will copy the framework from local repository (or) .git folder to github global repository.

Import + Clone URI:

Import will get the Framework from global repository to Local system.

Clone URI will create a folder in the same name as the frame work URL in local system.

pull:

pull will help us to get the changes from the global repository.
Only imported projects can be pulled.
We can use pull only if the project is existing in working directory.

Merge:

It will merge one branch at a time with the master.

Fork:

~~If you~~ It will create a copy of repository from different account into your github account.

pull Request:

It is an intimation given by the Automation Tester who creates a branch to push the code to the manager (or) Team Lead saying a new branch has been created. Later manager will look into the branch, analyze the code, if there are no conflicts, he will review and merge it into master branch of the frame work.

Branching in github :-

Everytime a new Engineer joins the Team, he will not be given collaborator ~~Right~~ Write into the master branch of the framework. He will be given a separate branch where he can pull the framework and push his code to the branch. Later Senior tester will review it & Merge it into master branch.

- Branching avoids unnecessary conflicts
- It makes sure that the master copy is safe. from unreviewed changes.
- It avoids code ~~splitter~~ Written by new Engineers which might effect the existing framework.

Maven

- Maven is a Build Management Tool. (Processing) involves Build development, Deployment, Testing
- It is used by both Developers & Automation Test Engineers.
- Development Usage:
- It is used for Build processes like Build Testing, Build creation and Build Deployment.

Build Testing:

- Testing the compilation issues the entire project and provide message called Build Success (or) Build failure.

Build creation: (processing)

- The process of converting source code into executable format.

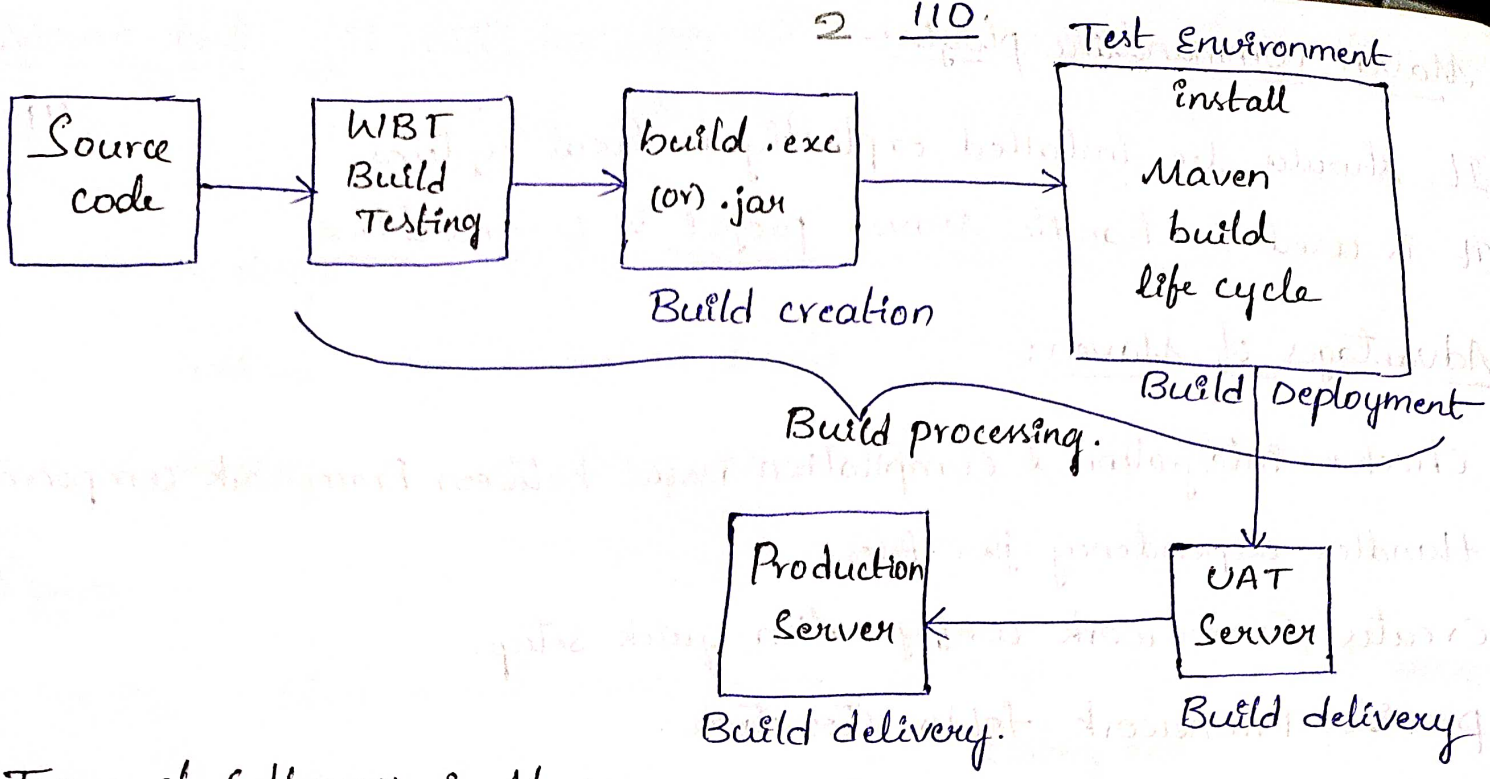
Build Deployment:

- The process of installing the Build into Testing server (or) any other servers.

Testing Usage:- check integration b/w frame work components & to execute Test scripts

In Automation Testing, Maven is used for Frame work build Testing and Test script execution.

- Multiple Engineers work with the same Frame work, there might be chances, one Engineer modification effects entire frame work, In order to avoid this, we go for Maven.



Types of Softwares in Maven:-

→ Maven has 2 types of softwares.

1. Maven IDE plugin
2. Maven Commandline plugin

Maven IDE plugin:

It is a default plugin available in Eclipse IDE used to create Maven project in Eclipse.

It provides default folder structure for the project.

Folders in Maven:

Page object Model

src/main/java → generic libraries, object Repositories (POM Pages)

src/main/resources → driver executable files

src/test/java → Test scripts

src/test/resources → external files to store Test data

(Project object Model)

pom.xml → Heart of maven project (Dependencies & plugins)

target → Test results

test-output → failed test scripts in xml, testng-results.xml reports

Maven Commandline plugin:- 3 Execute all the test scripts

111

- It should be installed explicitly in local systems.
- It is used to handle Maven project in command line.

Advantages of Maven:-

- Checks integration & compilation issues between Framework components.
- Handles dependency jar files.
- Creates Framework configuration quick setup.
- provides Framework folder structure.
- We can run Test Scripts in command line without Eclipse.
- It majorly supports jenkins.
- It supports commandline parameters.
- It also supports profiling.

What is dependency in Maven ?

It is advanced feature in Maven which is used to get all the required jars from global repository to Local repository. (.m2 folder) and attaches them to the project automatically.

Maven Build Lifecycle:-

Maven clean: It is used to clean all the old reports from target folder.

Maven validate: It is used to check if all the necessary jars are downloaded for all the dependencies added, if not, it will automatically download.

Maven compile: It is used to check the compilation ~~issues~~ issues in the framework.

Maven test: It will identify all the test classes whose classname is suffixed with test and executes them.

mvn package ⟶ Build creation

mvn install ⟶ to install build.

mvn display ⟶ to deploy the build into any other server.

Syntax :-

- mvn test -Dtest = <testclass.java> ⟹ to run specific test from testng.xml.

- mvn test -Dtest = <testclass1.java>, <testclass2.java>,

  to run more than one test script in testng.xml.

- mvn test -Dtest = <testclass.java> # <method name>

  to run specific @Test method in given Test class.

- mvn test -Dtest = <testclass1.java> # <method name>,
  <testclass2.java> # <method name>, ----

  to run multiple specific @Test method in given Test class.

⇒

Maven parameterization :-

Syntax :

- mvn test -Dkey1 = value 1 -Dkey2 = value 2.
- Passing the parameters through maven command line & performing Test execution is called maven parameterization.

Ex :- mvn test -Durl = https://actitime Demo. login.
-Dtime = 10 -Duser = admin -Dpwd = manager.

- In program the parameters can be accessed during runtime using "System.getProperty ("key")"
- All the values in maven parameterization will be always in "String" datatypes.

Maven Profiling :-

- Sure-fire plugin allows us to read testng.xml via pom.xml.
- If we want to execute specific suite file, Everytime we need to make changes in surefire plugin in pom.xml.
- This is not recommended because making so many changes to pom.xml might corrupt it. so if pom.xml is corrupted we have to discard the entire project.
- To avoid this, we will create a separate profile for every suite xml file in pom.xml & we will call specific profile by its profile ID in maven command line execution.

Syntax: mvn test -P <profile name/Id>

Structure:

```
<profiles>
  <profile>
   <id> profile name1 </id>
   <build>
      Surefire. plugin with xml file name.
   </build>
  </Profile>
  <profile>
   <id> profile name 2 </id>
   <build>
      Surefire. plugin with xml file.
   </build>
  </profile>
</Profiles>
```

Jenkins:

- Jenkins is CD/CI Tool used by developers, devops and Automation Testers.
- Basically jenkins automates:
  - process of Build creation → continous development
  - process of installing build into Testing environment → continous deployment
  - process of checking integration issues between old feature and new features → continous integration
  - process of delivering the tested build to the production environment → continous delivery.

Why jenkins in development:

- It is used for continous development, continous deployment and continous delivery.
- It continously monitors the git source code repository and creates a new build.
- It gets the latest build from git location and deploys into Testing server.
- Once the build is tested, it is deployed to UAT or production environment.

Why jenkins in Automation Testing:

- Automation T.E. use jenkins for continous integration
- The test scripts are executed continously to check integration issues between old build and new build.
- It is helpful to know the impact of new feature over the old feature build.

Types of Scheduling in Jenkins :-

1. On Demand :

Whenever the customer demands the execution, we trigger the execution with "Build Now" option. This is called on Demand Scheduling.

2. On Schedule :

Specifying the exact time duration of when the execution should start is called On schedule.

- While configuring the job, we have to select "Build periodically" in "Build triggers". and specify the time in following format.

Minute	Hour	DOM (Day of Month)	Month	DOW (Day of week)
(0-59)	(0-23)	(1-31)	(1-12)	(0-7) → 0 & 7 are Sunday

(Star)
* → any.

3. Poll SCM (Source code Management) :

- It will detect the changes made in specified git location and it starts the execution on scheduled time.
- Whenever any changes are detected like new push/commit, it will trigger the job.

→

Jenkins parameterization:-

- Jenkins also supports parameters but internally it is Maven parameters only.
- We use "$" for specifying jenkins parameters. and then in 'Goals & options' Maven command is provided as follows:

```
test -Dkey1 = $value1  -Dkey2 = $value2 ...
```

  - key1 & key2 are used to fetch values using System.getProperty() in the script.

Build pipeline in jenkins:

- Creating dependency between Multiple jobs and executing them sequentially is called pipe lining.
- The previous job is called upstream job
- The next job is called Down stream job.
- If one job is triggered, it will be executed and the status of this job will automatically trigger the next job.
- To perform pipelining in jenkins, we should integrate jenkins with build pipeline plugin. This plugin helps us to see the pipeline views where all the jobs which are in pipeline are visible at once.

Advantages of Jenkins:

- It monitors the build and checks integration issues of build.
- It supports 3 types of executions / job schedulings.
- It supports Run time parameters.
- It also supports pipelining of the jobs.
- It sends out an execution report via E-mail, once the execution is completed.

pre-requisites for jenkins:-

1. JDK-17 installation
2. apache maven -3.9.4 Zip download
   - path settings for Environment Variables
3. git download & installation.

Jenkins installation & configuration:-

1. Jenkins download
2. jenkins.msi → installation
   1. Run service as local system
   2. JDK 17 Path
   3. Test port (8080)
3. Setting admin password
   - copy pasted the specified path in New tab & fetched the administrator password.
4. Clicked on "Install Suggested plugins"
5. Created User Credentials.

6. Logged in to jenkins using same credentials

7. Installed required plugins.

- Go to Manage jenkins → under system configuration → Go to Manage plugins / plugins

1. Maven Integration
2. Github Integration
3. Build pipeline
4. HTML publisher

} Searched in Available plugins & clicked on "Install without restart"

8. Path settings

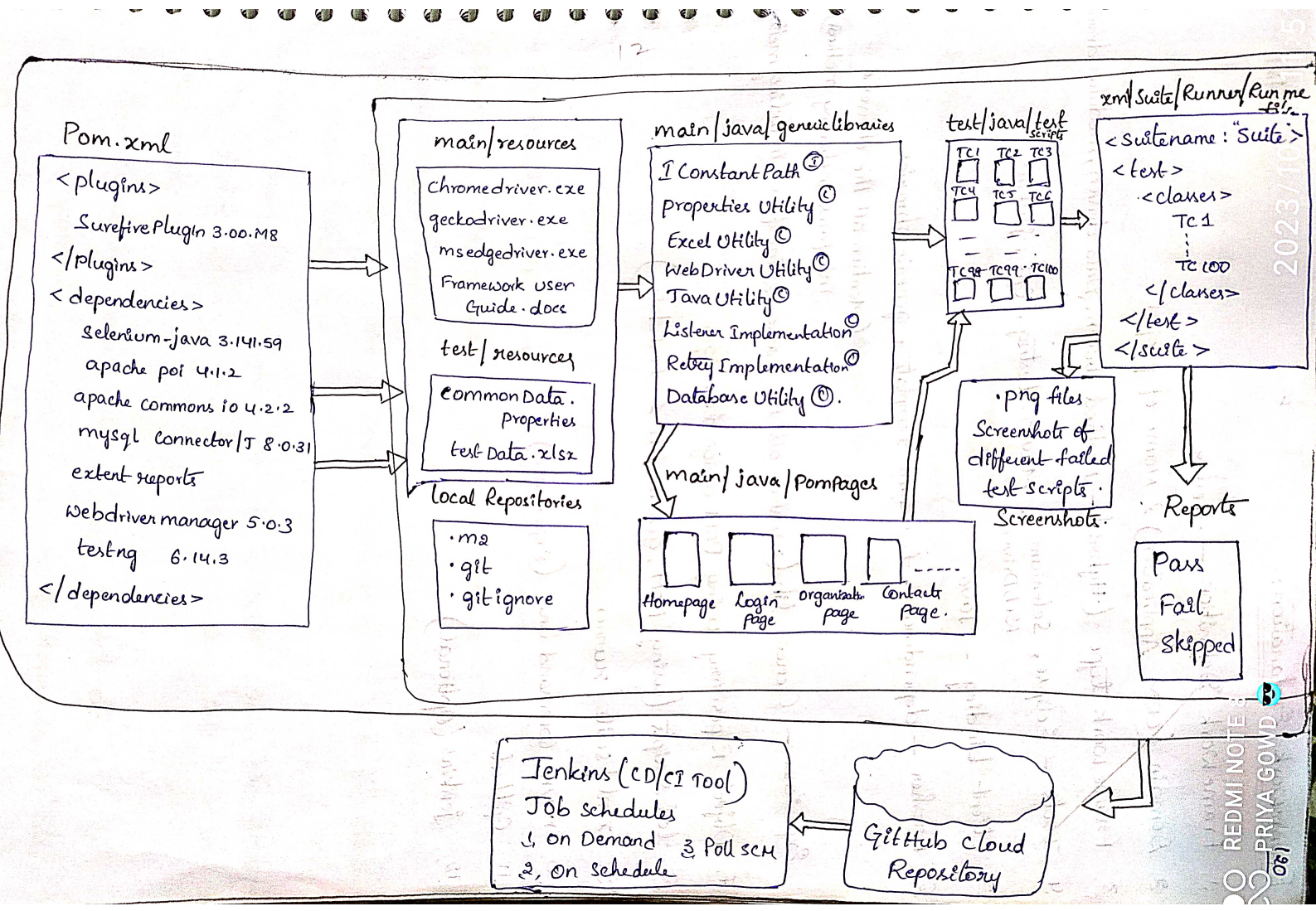
- Go to Manage jenkins → under system configuration → Go to "Tools"

1. JDK-17
2. Git
3. Maven 3.9.4

} Available in C:/Programfiles/.

Create & configure job in jenkins:-

1. Click on "New Item".
2. provide job name, select 'Maven project' → Click on 'ok'
3. (Automatically configure page is displayed)
3. under SCM, click on 'git' & provide github base url of framework Repository.
4. → under Build → Goals & options → provide 'test' (mvn command)


Pom.xml
<plugins>
Surefire Plugin 3.00.M8
</plugins>
<dependencies>
Selenium-java 3.141.59
apache poi 4.1.2
apache commons io 4.2.2
mysql connector/J 8.0.31
extent reports
Webdriver manager 5.0.3
testng 6.14.3
</dependencies>
main/resources
Chromedriver.exe
geckodriver.exe
msedgedriver.exe
Framework user Guide.docx
test/resources
Common Data. Properties
test Data.xlsx
local Repositories
.m2
.git
.gitignore
main/java/genericlibraries
I Constant Path
Properties Utility
Excel Utility
WebDriver Utility
Java Utility
Listener Implementation
Retry Implementation
Database Utility
main/java/Pompages
Homepage
Login page
organization page
Contacts page.
test/java/test scripts
TC1 TC2 TC3
TC4 TC5 TC6
TC98 TC99 TC100
xml/Suite/Runner/Run me
<suitename: "Suite">
<test>
<classes>
Tc 1
Tc 100
</classes>
</test>
</suite>
.png files
Screenshots of different failed test scripts.
Screenshots.
Reports
Pass
Fail
Skipped
Jenkins (CD/CI Tool)
Job schedules
1, on Demand 3. Poll SCM
2, On schedule
GitHub cloud Repository

Framework Explanation :-

1. Framework Definition — fresher
2. Domain (CRM Application) & Name of project
3. Framework type (Hybrid - method driven, Data driven, modular driven)
4. Tools - Eclipse, Selenium, Maven, testng, Git, jenkins.
   IDE — WebDriver — test automation tool — build management tool — Unit testing tool — Version control tool — CI/CD tool
5. Client binding — Java (Selenium-Java)
6. Fresher — dependencies, test data template, generic libraries.
   - Properties file
   - Excel file

   (Mention all the classes & interfaces including methods)

   Experience — Generic lib (mention all the classes & interfaces including methods)
7. Object Repository using POM design technique.
8. test scripts (using POM, generic lib & test data)
9. xml / suite / runner
10. mvn command line (Experience)
11. Github
12. jenkins (Optional)

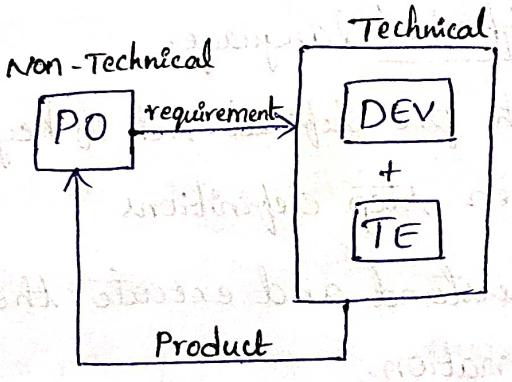
Frame Work:

It is a well organized structured collection of Reusable components such as Generic libraries which facilitates development, execution and modification faster and easier.

Frame work Approaches:

1. TDD (Test Driven Development)
2. BDD (Behaviour Driven Development)

TDD (Test Driven Development)



Priority — Technical aspects

→ In Traditional development, the priority is given for technical aspects such as skills, Architecture, Resources, softwares etc...

→ Communication among the technical & non-technical members is not enforced. so, the chances of developing the wrong product is comparitively more.

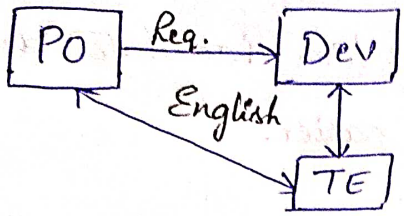
→ Here, manual T.E. develops Test case document only then Automation testers develops the Test scripts looking into Test case document.

→ Tools used here are testng, j-unit, n-unit etc;

In this approach, we will have lot of reusable methods.

Test the driving factor here.

BDD (Behaviour Driven Development):



→ In this approach, the communication among the Team members is given the first priority.

→ chances of developing wrong product is less.

→ Here, Automation Testers will develop feature files which act as manual test case document.

→ feature files are developed using 'Gherkin' language.

→ It is normal English language with pre-defined set of keywords.

→ All the steps in feature file are given step definitions

→ Even Non-technical person can understand and execute the feature files to perform Test Automation.

→ Tools used in BDD — Cucumber, Specflow, jbehave etc;

→ feature files are the driving factor here.

→ we will have less reusable methods.

Types of Frame Works:-

Data driven

Method driven

Modular driven

keyword driven

Hybrid.

Data driven Framework:

- Reading the data from any external resources (or) using data providers is called as Data driven Framework.
- Data is the driving factor.
- Whenever test data is huge compared to test scenarios, we prefer Data driven framework.

Ex:- E-commerce, Banking, Finance, travel....

Method driven Framework:

when an app has lot of functionalities

- Developing the reusable methods for all the repititive actions (or) functionalities in the applications and utilizing these methods in Test Scripts is called as Method driven Framework.
- Methods are driving factor.
- Whenever the application contains more repeated functionalities like too many ~~problems~~, dropdowns, frames, windows etc; we go for Method driven framework

Eg: E-commerce, CRM, Health care....

Modular driven Framework:

- Maintaining the test scripts, test data and suite xml files module wise in order to ~~be~~ make the debugging process easy is called as modular driven F.W.
- Modules are the driving factor.
- Whenever the application is very huge and has lot of modules, we prefer modular driven F.W.

Eg: CRM, SCM (Supply chain management), E-commerce, ERP (Enterprise Resource

Keyword driven Framework:

- Creating the keyword library and utilizing these keywords to develop the test scripts is called as Keyword driven F.W.
- Keywords are the driving factor.
- Whenever manual testers (or) freshers have to perform automation, they would not be good at coding, so, all the actions are converted into methods with user friendly names, so that they can perform the actions easily.

Eg: Short term projects like Education, job portals.

Hybrid Framework:

- Combination of two (or) more Framework types is called as Hybrid Framework.
- Possible Combinations — Datadriven + Method driven
  Data driven + Modular driven
  Data driven + Method driven + Modular driven
  Data driven + keyword driven
  Data driven + keyword driven + Method driven.

Ex: All the long term projects like E-commerce, CRM, ERP, SCP.

Advantages of Framework:

Refer to document in github repository.

Stages / Phases of Developing Framework:

1. Framework choice

   Hybrid → Method Driven + Data Driven + Modular Driven.

2. Design

   → Adding dependencies to Pom.xml

   → Adding driver executable files → src/main/resources.

   → Adding testdata template like properties files & excel files
   ↓
   src/test/resources

   → Developing generic libraries / utility classes → src/main/java

3. Development

   → Developing POM classes → src/main/java

   → Developing test scripts → src/test/java

4. Execution

   → Batch / Group / Parallel execution using xml / suite files.

   → Maven cmd line execution / Jenkins execution

   → Report generation.

BDD Framework :-

→ BDD is a software development approach which allows the tester (or) Bussiness Analyst to create Test cases in simple text language (English)

→ BDD inherits all those features present in TDD along with it's advantages.

- Test Scenarios are written separately in feature file.
- Tests are written by focussing user stories and system behaviour in 'Layman' language.
- code is written in step definitions file using programming languages.

→ BDD acts as a bridge to overcome the gap between technical and Non-technical teams.

→ This helps and improves communication among technical and Non-technical teams, managers, stake (stake) holders.

Tools used to build BDD Frame work:

Cucumber (java, python, javascript, .Net, Php, Ruby, pearl etc....)

Specflow (.Net)

Jbehave (java)

Mocha (javascript)

Kahlan (Php)

Lettuce (Python)

Cucumber:

→ It is an open source tool which is widely used to develop BDD framework.

→ Using cucumber, we develop the test cases for the behaviour of software functionalities.

→ It is mainly used to write acceptance test script for Web Applicat-ons

→ The test cases in cucumber are written in 'Gherkin' language

→ Gherkin is the language parser used in cucumber which is a simple text in english with predefined keywords.

Gherkin keywords:

1. Feature — High level description of feature to be tested.
2. Scenario — It is used to define the test case steps.
3. Given — used for precondition
4. When — used for actions to be performed.
5. Then — used for validations (or) outcomes of the Test cases.
6. And (or) But — Used as conjunctions along with given when and Then.
7. Scenario Outline — used for data driven testing.
8. Examples

~~Examples:~~

App like data providers we provide multiple sets of data to be tested

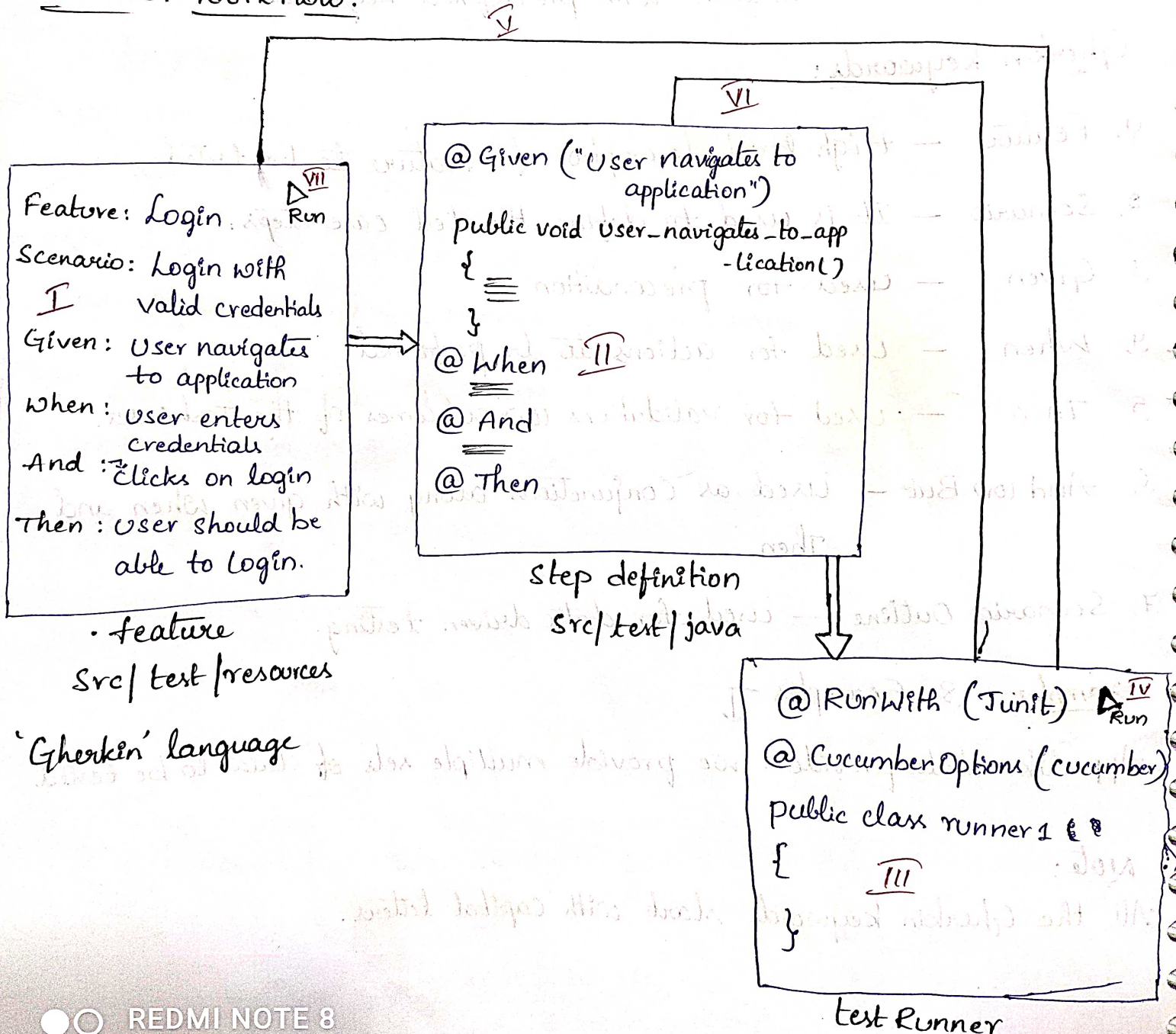
Note:

All the Gherkin keywords start with capital letters.

Pre-requisites to implement Cucumber:

1. Install cucumber plugin in Eclipse.
2. Create Maven project.
3. Dependencies to be added are cucumber-java
   cucumber-junit
   cucumber-reporting
   selenium-java
   Guava

Cucumber Workflow:



Reports:

- In Cucumber, we can generate different reports like html, Junit and Json reports.
- The pre requisite to generate reports is adding the dependency called Cucumber reporting.
- In order to generate the reports, we have to add plugins to TestRunner file within @CucumberOptions.

⇒

```
@RunWith (Cucumber.class)
@CucumberOptions (features = "<.feature path>",
                  glue = "stepdefinations",
                  monochrome = true,
                  plugin = { "pretty",
                             "html:<html report path>",
                             "junit:<junit report path>",
                             "json:<json report path>" }
public class TestRunner {

}
```

monochrome = true:

It helps to generate report in Human Readable format

pretty:

It ~~helps~~ is a plugin which helps to align the report in proper format.

Background:

- It is used to setup a precondition.
- It is executed before each scenario but after @Before hook.
- We should add Background before the first Scenario in the feature file.
- Background is provided when we wanted to give Customer Readable preconditions to the scenarios.

Hooks:

- Hooks are the blocks of code that run before (or) after each Scenario (or) step.
- They (Hooks) allow us to manage the code workflow.

Types of Hooks:

@Before → Executes before every Scenario

@BeforeStep → Executes before every step in feature file.

@AfterStep → Executes after every step in feature file.

@After → Executes after every Scenario.

- We use Hooks when we have to do some technical setups before and after scenarios.

Order of Execution - Hooks vs Background:

@Before → Background → Scenario → @After

Tags:

- In Order to group similar kind of test cases, cucumber provides the feature called tags.
- tags are normally used over scenarios within the feature file and also over the features
- The name of the tags are user defined

Example:- @integration → Feature level tag

```
Feature : Sample2
@smoke @regression
Scenario: S1
@system @important
Scenario: S2
@important
Scenario : S3
```

} Feature file

- Test Runner for Tags

```
@RunWith (Cucumber.class)
@CucumberOptions (features = " ", glue = " ",
                  tags = "@smoke or @integration")
Public class TestRunner {
}
```

→ We can provide multiple tags for one Scenario/feature.

→ To specify which tag scenario has to be executed, we should provide tags in CucumberOptions.

Multiple tags can be executed by giving those tag names using the conjunctions 'and', 'or', 'and not', 'or not'.

TestNG with Cucumber:

1. TestNG plugin
2. Dependencies
   - Cucumber java
   - Cucumber testng
   - testng
   - selenium java
   - cucumber reporting
3. Test runner class

```
@CucumberOptions(features="  ", glue="  ")
Public class TestRunner extends AbstractTestNG ng CucumberTests.
{
}
```