Android – OS

Iphone – OS

Installation Steps for WebDriver

Installation is divided into two parts.

1. Download WebDriver jar from comunity

① Google → Download Selenium

Link on first link & navigate to Selenium community

② Find Selenium client & WebDriver language Bindings division & click on download link beside the java (2.47.1)

↓ latest version

③ Download zip folder & place it in C drive

Size of Selenium jar → 21MB

④ Extract zip folder

Import all WebDriver jars to Eclipse

① Open Eclipse with new workspace

② Create a new java project

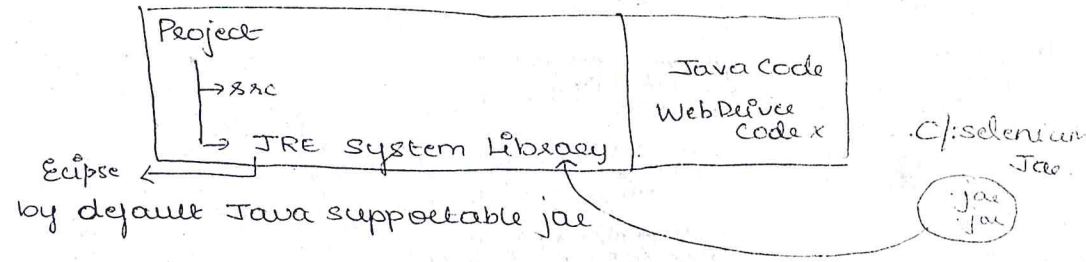
File → New → java project

Eg Project Name : Selenium Project

* Make sure while creating project JRE should point to latest version (jdk 1.8.0_25 or 1.7)

③ Create a new package along with java class

Eclipse has default java

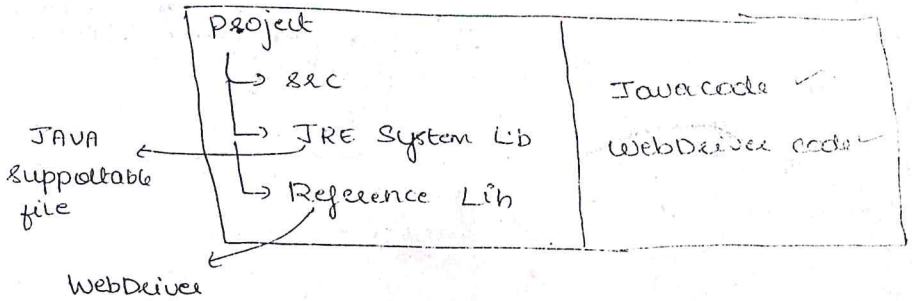


④ Select a project, right click, select Build path option & click on Configure Build path.

⑤ Click on Libraries Tab in Property window

⑥ click on Add External JAR's button

⑦ Import all the WebDriver jars in C-drive



NOTE

* Make sure all the jars available inside Lib's folder and present along with Lib folder should be imported to Eclipse IDE.

Sample Code

```
package pac ;
import org.openqa.selenium.WebDriver ;
```

```
import org.openqa.selenium.firefox.FirefoxDriver;
public class MyFirstWebDriverCode
{
    public static void main(String[] args)
    {
        WebDriver driver = new FirefoxDriver();
        driver.get("http://gmail.com");
        driver.quit();
    }
}
```

→ When we run this it only launches the browser, & removes the plugin that found in we launched browser (Diff b/w selenium & we launched browser)

Locators

Locators are used to identify any webelement in UI based html source code

By — Abstract class

- id();
- name();
- xpath();
- CSS-selector();
- LinkText();
- partialLinkText();
- ClassName();
- TagName();

All are static methods

All these locators are static method

Writing WebDriver test scripts — Typically uses 5 steps

1. Launch empty browser
2. Navigate to any web based Application
3. Find a Web element in UI
4. Perform operation on web element
5. close browser.

E.g

```
public class MyFirstWebDriverCode
{
    public static void main(String[] args)
    {
        //step1 : Launch empty browser
        WebDriver driver = new FirefoxDriver();

        //step2 : navigate to gmail App
        driver.get("http://gmail.com");

        //step3 : find (userName editbox) webel
                  in UI
        WebElement wb = driver.findElement(
                        By.id("Email"));

        //step4 : perform operation on editBox
                  WebElement.
        wb.sendKeys("qspiders.selenium@gmail.
                     com");
```

```
//step5 : close browser
driver.quit();
}
```

→ WebElement wb = driver.findElement(By.id("Email"));

- → returns WebElement, if element available in UI
- → throws "No such Element Exception", if element not present in UI

→ To import, type WebD → and press Cntrl+space

type Firef → Cntrl+space.

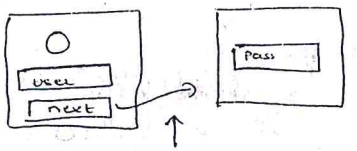
(or)

type WebDriver and space cursor on it & then we can import

Write a code for Gmail Login & Logout test?

→

```
public class GmailLoginLogoutTest
{
    public static void main(String[] args) throws InterruptedException
    {
        WebDriver driver = new FirefoxDriver();
        driver.get("http://gmail.com");
        driver.findElement(By.id("Email")).sendKeys("qspiders.selenium");
        driver.findElement(By.id("next")).click();
        Thread.sleep(2000); --- 2s (it is written in ms)
        driver.findElement(By.id("Passwd")).sendKeys("qspiders");
        driver.findElement(By.id("signIn")).click();
        Thread.sleep(7000);
        driver.findElement(By.xpath("//a[contains(text(),'@gmail.com')]")).click();
        Thread.sleep(1000);
        driver.findElement(By.xpath("//a[text()='Sign Out']")).click();
    }
}
```



↑ it has to wait for some seconds

So we are adding a delay

Thread.sleep(2000); → java code is added

If not written it throws exception (No such Element

NOTE

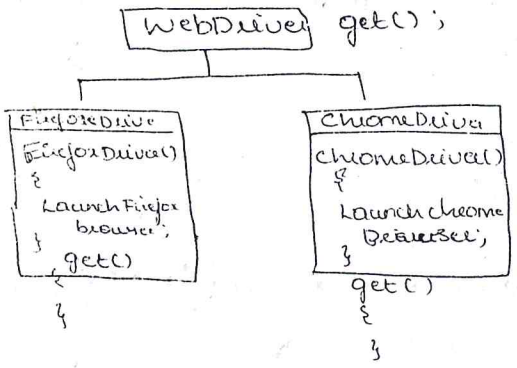
* whenever WebDriver launches the browser, all the browser plug-ins automatically getting disabled
* WebDriver launches new browser for every execution.

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

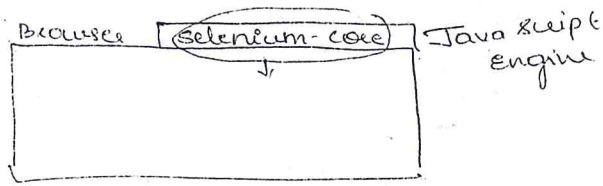
When these instruction executed we get ~~new~~ new browser?

→ When an object is created, the default constructor will execute Hence All browser have a constructor to launch new browser

* Whenever we create a object to FirefoxDriver class by default empty browser will be launched, because all browser class constructor has code to launch Specific Browser



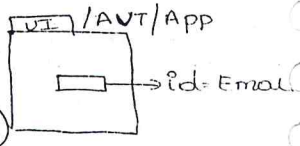
* Selenium Core is a java script Engine comes along with WebDriver jar, which is used to execute WebDriver commands on the Specific browser



* For automatic package import, better practice to use Cntrl + Space

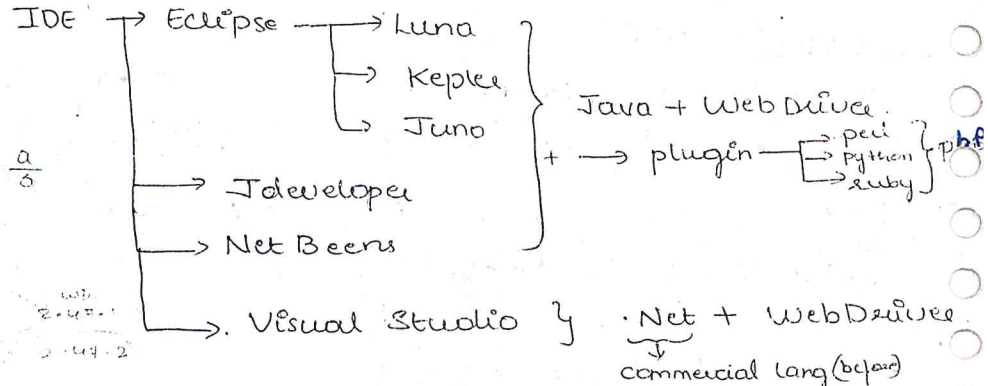
* findElement() method is used to identify the WebElement based on 8 locators which returns WebElement if object available in UI Or else throws "NoSuchElement" Exception

Eg
```
driver.findElement(Locators)
driver.findElement(By.id("Email"))
```



Problem Areas

* While working with WebDriver, better practice to use latest IDE (Integrated Development Environment)



* Whenever we download latest version of WebDriver jars from Selenium community, better practice to work with latest browser

WebDriver has backward comptability but no forward comptability

NOTE

* Always better practice to disable automatic update in Firefox Browser

Go to Tools → Option → click on Advanced tab
↓
Update & select
Never check for updates

* In order to open Eclipse, Jdk should be installed with path Environment Variable

Jdk → 64 bit → set path
Eclipse → 64 bit

* While creating a java project in Eclipse, Jdk compiler should point to latest version
Build path → Java Compiler.

Code optimization.

* Getting same output with ~~diff~~ set of statement is called code optimization.
* Getting same output with less number of statements is known as code optimization

Eg

```
int i = 10;
int j = 20;
int z = i+j;
Sop(z);
```

```
int i = 10;
int z = i+20;
Sop(z);
```

```
int z = 10+20;
Sop(z);
```

Sop (10+20);

Output
30

Eg

```
class F
{
    public void fe()
    {
        System.out.println("execute fe");
    }
}

class W
{
    public void sendKeys()
    {
        System.out.println("execute sendkeys");
    }
}

public class Run
{
    public static void main(String[] args)
    {
        F f1 = new F();
        W w1 = new W();
        f1.fe();
        w1.sendKeys();
    }
}
```

Step 1

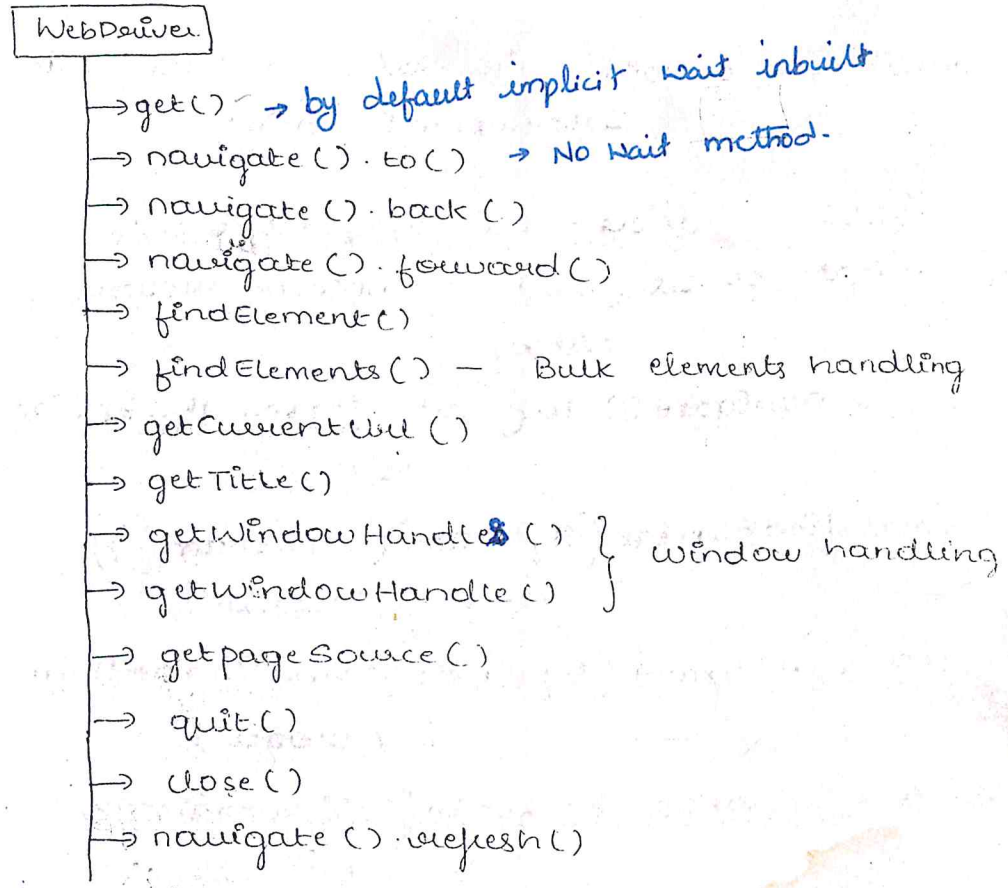
```
class F
{
    public W Ref()
```

```
    }
    System.out.println("execute fe");
      W w1 = new W();
      return w1;
    }
}

class W
{
   public void sendKeys()
     {
       System.out.println("execute sendkeys");
     }
}

public class Run
{
  public static void main(String[] args)
    {
      F f1 = new F();
    W w1 = f1.get();
      w.sendkeys;
    }
}
```

Step 2

```
F f1 = new F();
f1.get().sendkeys();
```

WebDriver Basic API's



All these methods are present in WebDriver interface, All these methods are implemented by all browser classes

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

Program

```
public class WebDriverBasicMethods
{
    public static void main(String[] args) throws
                                    InterruptedException
    {
        WebDriver driver = new FirefoxDriver();
// navigate to url(app), by holding browser
                                    history
        driver.navigate().to("http://qsp-ab-as-pc/
                                    login.do");
        driver.findElement(By.name("username")).
                                    sendKeys("admin");
        driver.findElement(By.name("pwd")).sendKeys
                                    ("manager");
        driver.findElement(By.xpath("//input[@type=
                                    'submit']")).click();
        // click on browser back button
        driver.navigate().back();
// click on browser forward button
        driver.navigate().forward();
// refresh current page
        driver.navigate().refresh();
// maximize the current active browser which is
                launched by WebDriver
        driver.manage().window().maximize();
        // delete all browser history
        driver.manage().deleteAllCookies();
// close all windows launched by WebDriver
        driver.quit();
    }
}
```

Web Element basics API's

- Web Element
  - sendKeys()
  - click()
  - clear()
  - isEnabled()
  - isDisplayed()
  - isSelected()
  - getTagName()
  - getLocation()
  - getCSSValue()
  - getText()
  - getAttribute()

* Any element available on webpage like editbox, checkbox, radio button etc called Web Element

* All the methods available in web element interface used to perform specific action on web element (like click, type etc...)

* Before performing operation on any web Element, Selenium has to uniquely identify the web element using findElement() method.

Eg:

Verify 'invalid message [-ve TC]

Step① navigate to Gmail App

Step② click on login button without UN & Pwd. Expected result "username pwd invalid please try again" should be displayed.

```
public class WebElementBasicMthods
{
  public static void main(String[] args) throws
                                InterruptedException
  {
    String expectedMsg = "Username or
                          password is invalid";
    // navigate to app
    WebDriver driver = new FirefoxDriver();
    driver.get("http://demo-pr.in/login.jsp");
    // click on login without data
    driver.findElement(By.id("loginButton")).
                          click();
    Thread.sleep(3000);
    String xpathmsg = "//span[contains(text(),
                          'username')]";
    WebElement wb = driver.findElement(By.
                          xpath(xpathmsg));
    // verifying text webelement in UI
    String Actmsg = wb.getText();
    // it captures the visible text from webelement
                          returns String
    if(expectedmsg.equals(actmsg))
    {
      System.out.println("msg is verified == Pass");
    }
    else
    {
      System.out.println("msg is verified== Fail");
    }
    driver.quit();
  }
}
```

Test Case name :-

Verify Gmail Login page

Step1 : Navigate to Gmail App

Step2 : Verify the Gmail Login page features

Expected Result

① username editbox should have default value 'Enter your Email'

② check box should be ~~was~~ checked by default

③ Gmail login should have Google logo image

④ Forgot password link should be displayed.

```
<input (id) 'Email' (name) = 'Email' (placeholder) =
'Enter your Email' />
```

```
<a (id) 'link - forget password' href = '...' > Forget
password </a>
```

getAttribute()

↓

used to get the values of the back end attribute

getText()

↓

used to get visible text if present, returns string or else returns empty string.

① d: fE(By.id("Email"))

↳ getAttribute("placeholder")

↳ String.

② d: fE(By.id("Link ...")) → getText()

① ☑ To check the radiobutton / checkbox is already selected we use isSelected() method

↳ boolean

To check any string is displaying in page, we use.

isDisplayed() method

↳ boolean

Example :-

```
public class VerifyGmailLoginPage
{
    public static void main(String[] args) throws
                                    InterruptedException
    {
    // test Data
    String expEditBoxVal = "Enter your Email";
    String expLinkName = "Forgot Password";
    String userName = "qspiders.selenium";

    WebDriver driver = new FirefoxDriver();
    driver.get("http://gmail.com");

// Step1 : userName edit box should have default
                                    value
    String atVal = driver.findElement(By.id("Email"))
                                    .getAttribute("placeholder");
    if(expEditBoxVal.equals(atVal))
    {
```

```
System.out.println(actVal + " = Value is Verified
                    => PASS ");
}
else
{
System.out.println(actVal + " = Value is not Verified
                    => FAIL ");
}

// step 4 : Verify forGot Password Link
driver.findElement(By.id("Email")).sendKeys(user
                    Name);
driver.findElement(By.id("next")).click();
Thread.sleep(9000);
String actLinkName = driver.findElement(By.id
        ("link-forgot-passwd")).getText();
if(expLinkName.equalsIgnoreCase(actLinkName)
{
System.out.println(actLinkName + " = Link
        Name is Verified == PASS");
}
else
{
System.out.println(actLinkName + " = Link
        Name is not Verified == FAIL");
}

// step 3 : google Logo image should be displayed
boolean imgstat = driver.findElement(By.xpath
    ("//div[@class='logo logo-w']")).
                    isDisplayed();
if(imgstat)
{
System.out.println("Google Logo is displayed
                    == PASS");
}
else
{
System.out.println("Google Logo is not
                    displayed == FAIL");
}

// step 8 : Check box should be checked by default
boolean chBoxStat = driver.findElement(By.id
                    ("PersistentCookie")).isSelected();
if(chBoxStat)
{
System.out.println("CheckBox is checked ==
                    PASS");
}
else
{
System.out.println("CheckBox is not checked
                    == FAIL");
}
```

```
    driver.quit();
  }
}
```

Output-

Enter your Email = Value is Verified == PASS

Forgot password ? = Link Name is verified == PASS

Google Logo is displayed == PASS

Check Box is displayed == PASS

NOTE

getText()

In order to capture text present within open and closed html Tag (visible text) we will go for getText() method, which always return String value if web element has visible text or it return empty String.

getAttribute()

In order to capture back end attribute available within html Tags angle brace, we go for getAttribute() method, which returns String value.

isDisplayed()

In order to check whether any web Element is displayed in UI, we go for isDisplayed() method, which returns boolean value (true) if web element is available in UI

isEnabled()

In order to check whether Object is enabled (active) we go for isEnabled() method, which returns boolean value.

isSelected()

isSelected() method is used to check whether radioButton or Checkbox is already selected or not, which returns boolean value (true) if checkbox / radiobutton is already selected / checked.

Assignment

Verify facebook Login page

① Navigate to fb

② verify the login page web Elements

Expected result

① Keep me Login checkbox should be unchecked in default

② First name edit box shld have default val

③ Login page shd have Create a page

(5) login page should have connect people image

```
public class VerifyFacebookLoginPage
{
  public static void main(String[] args)
  {
    String expEditBoxVal = "First name";
    WebDriver driver = new FirefoxDriver();
    driver.get("https://www.facebook.com");
    // step 1
    boolean actChBox = driver.findElement(By.
        id("persist_box")).isSelected();
    if(actChBox)
    {
      System.out.println("Keep Logged in Checkbox is
          unchecked == PASS");
    }
    else
    {
      System.out.println("Keep logged in checkbox
          is checked == FAIL");
    }
    // step 2
    String attval = driver.findElement(By.id("u_0_1")).
        getAttribute("aria-label");
    if(expeditBoxVal.equals(attval))
    {
      System.out.println(attval + " = value is
          verified == PASS");
    }
    else
    {
      System.out.println(attval + " = Value is not
          verified == FAIL");
    }
    // step 3
    boolean actLink = driver.findElement(By.xpath
        ("//a[contains(text(),'Create a Page')]")).
        isDisplayed();
    if(actLink)
    {
      System.out.println("Create a Page Link is
          displayed == PASS");
    }
    else
    {
      System.out.println("Create a Page Link is not
          displayed == FAIL");
    }
    // step 4
    boolean actimg = driver.findElement(By.xpath("//
        img[@class='img']")).isDisplayed();
    if(actimg)
    {
      System.out.println("image is displayed ==
          PASS");
    }
    else
```

```
}
    System.out.println(" image is not displayed == FAIL");
}
    driver.quit();
}
}
```

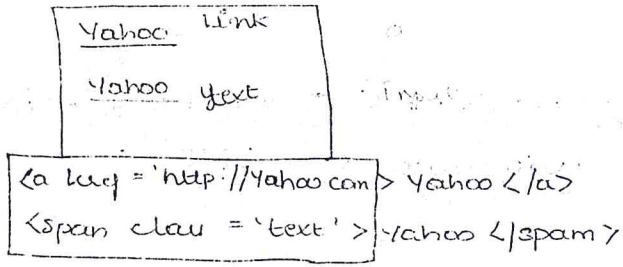
Output

Keep Logged in Checkbox is unchecked == PASS

First name = value is verified == PASS

Create a Page Link is displayed == PASS

Image is displayed == PASS



driver.findElement(By.xpath("//a[text()='Yahoo']"));

To identify it is a Link or not we use getTagName() ---> String tag = getTagName();

if tag == a ---> then it is a Link

getTagName()

getTagName() method is used to capture Tag from the web element

~~getCssValue()~~ | getCssSelector()

getCssValue() method is used to capture

Google [ ] [Search] ---> This button should be Blue

Use ---> getCssValue()

String actVal = getCssValue("colour")

↓

ASCII value

00xA ---> white

01xA ---> Red

10xx ---> Blue

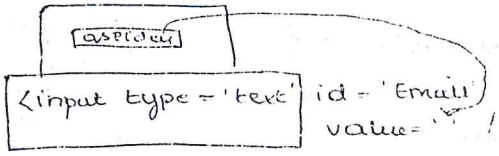
Interview Questions

1. How to clear the value in edit box ?

---> driver.findElement(By.id("Email")).sendKeys("qspiders");

driver.findElement(By.id("Email")).clear();

2. How to capture existing value in edit box ?



driver.findElement(By.id("Email")).sendKeys("qspiders");

String val = driver.findElement(By.id("Email")).getAttribute("value");

System.out.println(val);

O/P

qspiders

3. How to check whether the web element is displayed in UI ?

---> isDisplayed()

4. How to get colour of the object ?
→ getCssValue();

5. How to get backend attribute of the web element ?
→ getAttribute();

6. How to capture dynamic text visible in UI ?

moving text in a page [dynamic text] eventhough odv changes day by day but in html tag it remains same (from marque.

```
driver.get("http://ilecte.co.in");
String xp = "//marque/span/a[@href = 'http://www.
            ilectctourism.com']";
String text = driver.findElement(By.xpath(xp)).
              getText();
System.out.println(text);
```

Dynamic text
→ Identify proper xpath
→ getText()

WebDriver Wait Statement

- Normal Wait → Java wait
- Implicitly Wait
- Explicitly Wait
- page Load Wait

→ WebDriver wait

Statements are used basically after click();

* The process of matching Automation tool and with application is called Synchronization

→ If wait statements are not used, there may be failure (NoSuchElementException) due to loading of page takes time for that application.

1. Normal Wait

Syntax :- Thread.sleep("10000");
↓ milliseconds

E.g Thread.sleep("10,000")

* Normal wait always going to wait for specified amount of time.
* Normal wait is also called as Hard coded wait should not be used in real time Selenium script because whenever application download the elements within in 2s, Normal wait unnecessarily wait for another 8s

Gmail Login Logout

- → navigate to Gmail
- → Enter UN
- → click on next
- → NW(10) → 2s → (&) prob wastage
- → Enter pwd
- → click on LoginButton
- → NW(10)
- → Logout

Problems
→ modifications of script each time when app speed changes
→ Time consuming process

Instead we can do manually time based on above prob for Implicitly wait

Navigation from one page to another page → wait

3. Implicitly wait

Syntax :- driver.manage().timeouts().implicitlyWait(duration, type of duration);

* Implicitly wait, wait till page get download before performing any action on web element.

Eg driver.manage().timeouts().implicitlyWait(30, TimeUnit.seconds)

* Technically implicitly wait polling/monitoring entire html document before performing action on web element, if webpage get downloaded within 30s implicitly wait releases control to next line execution instead of waiting entire 30s

Disadvantage of Implicitly wait

(i) Implicitly wait cannot be used in AJAX application

(ii) Implicitly wait cannot wait for dynamic object.

To resolve these problem Explicitly wait is used

Connecting runs & stops & again starts connecting this implicitly wait when stops goes to next line for searching it throws NoSuchElementException

Explicitly wait

Syntax :- WebDriverWait wait = (obj, time)

→ until(ExpectedConditions)

presence of element located(locator)

In Ajax Application (&) dynamic object if Explicitly wait, if the webElement already found before Connecting then also it goes new link

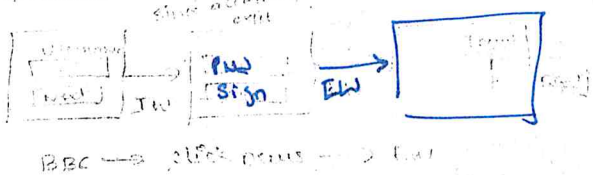
Eg WebDriverWait wait = new WebDriverWait(driver, 30);

wait.until(ExpectedConditions.presenceOfElementLocated(By.xpath("//a[contains(text(),'gmail.com')]")));

* Explicitly wait, wait till web element to be present in UI

* Technically, Explicitly wait check for the expected condition for every 500ms, if expected webelement appears within (0-30s), driver control release to next line execution instead of waiting entire 30 seconds.

* Explicitly wait plays a major role in Ajax application & dynamic object.



4. PageLoadTimeOut()

WebDriver driver = new FD();
driver.get("Gmail.com");

get method already have implicitly wait for unlimited about time.

pageLoadTimeOut() method is used to set time out for every action performed by WebDriver if page is not downloaded within specified time

Web driver throws an time out exception & stop the entire execution.

Eg

```
WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().pageLoadTimout(30, TimeUnit.SECONDS);
driver.get("http://gmail.com");
```

→ this throws an Exception since page is not downloaded within 1s

→ It is mainly used before get() method

→ Implicitly

Interview Question

1. How to handle dynamic object in WebDriver?
→ We go for explicitly wait, wait till object to be appear in UI before performing an operation.
2. How to identify dynamic Object?
→ contains() Ex following-sibiling:: function using advanced xpath function, we will identify.
3. Diff b/w Normal wait, Implicitly wait & Explicitly wait?

Normal wait	Implicitly wait	Explicitly wait
1. Hard coded wait time is fixed	1. Synchronization wait	1. Synchronization wait
2. Wait for specified amount of time	2. Wait for page to load before performing any action	2. Wait for webelement
3. Execution thread will sleep for specified amount of time	3. monitor Entire html document to load	3. Every 500 ms check for expected condition
Cannot be used in Ajax App		
Cannot use condition		

4. What are the Synchronization Wait available in wait statement
→ Implicitly & Explicitly Wait

5. Using FluentWait → we can customize explicitly wait.

Actions

Scenario's

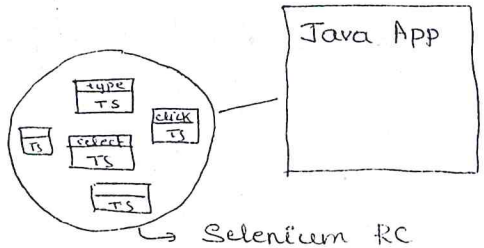
Step1: Navigate to FlipKart

Steps: Place mouse cursor Book & Media link (or) desktop menu bar

Steps: click on new release link below ebooks & eprints

Selenium RC

* Selenium is not a UI based & recorded & play back tool.
* Selenium RC is just a Collection of Java scripts method.
* Selenium RC introduced in 2004 by JASON
  Jr.
  Java Developer in Thought works



Advantage

* Application which it supports → web based Application
* Browser Support → firefox, Chrome, Safari IE, Opera
* OS/ platform Supports → Windows, Linux
* Lang supports → Java, Rubby, perl, python, php, C#

Disadvantage.

→ only support web based application
→ not a UI based tool & no recorded & playback tool
→ Selenium RC does not supports mobile

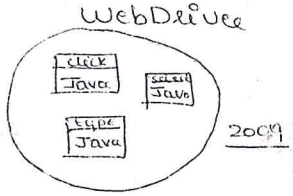
https:// → Secured network
(All device based are secured Network)

* Selenium RC is a collection of java script, so it is failed to work with secured networks

→ Secured network will not allow to execute java script on beause because of security.

* Selenium RC is not UI based so we use other source to write test cases (i.e) in eclipse

JASON in 2006 started working in Google

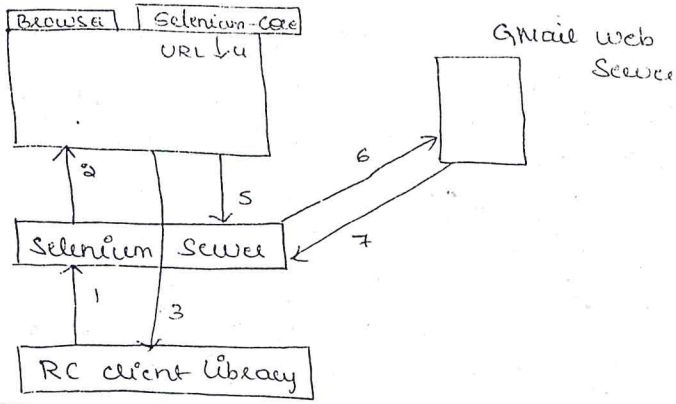


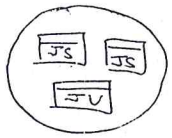
(Folder) collection of core java API source.

→ WebDriver Supports all secured networks

* App supports → web based Application
* OS/ platform Supports → windows, Linux, Mac-OS Android, IOS} mobile.
* It is a Open source tool
* Brouser Supports → firefox, chrome, Opera IE, safari
* Lang Supports → Java, Ruby, perl, python, php, C#
* WebDriver has Complete support with secured network.
* It is not UI based & Record & playback tool

Architecture of Selenium RC



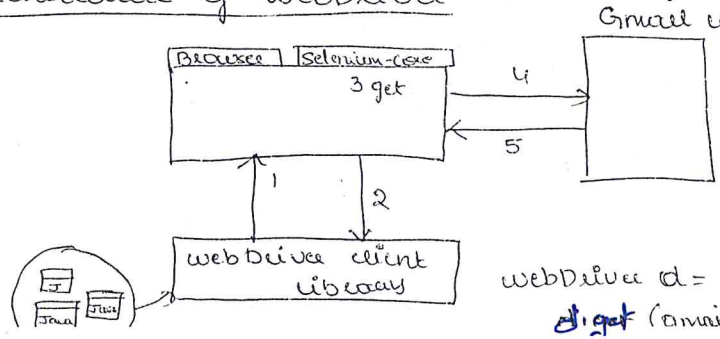


Selenium open (gmail.com)
... type ('x-path UN', 'admin')

Selenium RC is collection of JS when we want to open a brouser using selenium RC it first interacts this with Selenium server this will open brouser to the client then the selenium-core in Selenium server used to get url & then interacts the GMail web server to .gmail page.

Hence selenium RC uses selenium server as an intermediate b/w the client & web App so it is slower in execution.

Architecture of WebDriver



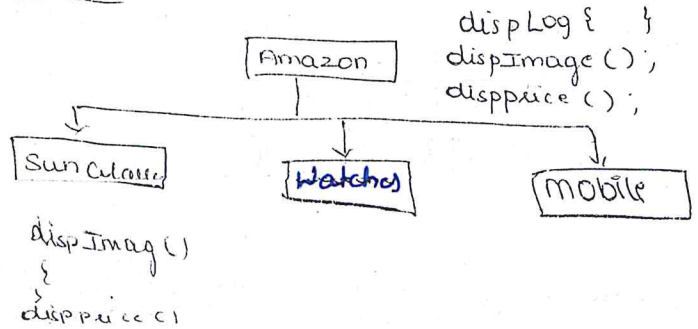
WebDriver d = new FD()
d.get (gmail.com)

WebDriver does not use intermediate server so it is faster in access

Difference between Selenium RC and WebDriver?

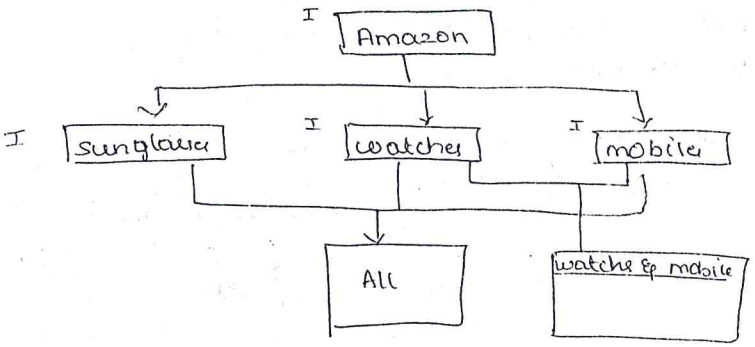
Selenium RC	WebDriver
* It a collection of java scipt API	* Webdriver API are completely object oriented
* Selenium RC cannot communicate to browser without server	* Webdriver can communicate to all browser without server.
* Selenium RC is slower in execution compared to WebDriver	* webdriver is faster in execution.
* Selenium RC does not supports mobile platform	* Web driver supports mobile platform
* In order to run Selenium RC test scripts, server should be up & running	* Web driver server is not required
* partial support with secured network	* Webdriver has complete support with secured network

Abstract class



Interface

100% Abstraction
& multiple inheritance is possible



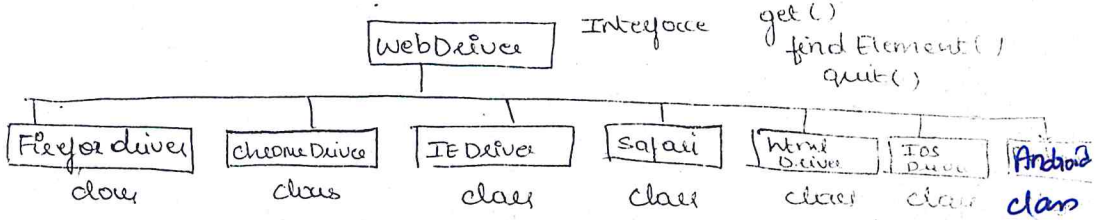
Run time polymorphism

Same method executes diff behaviour based on instance.

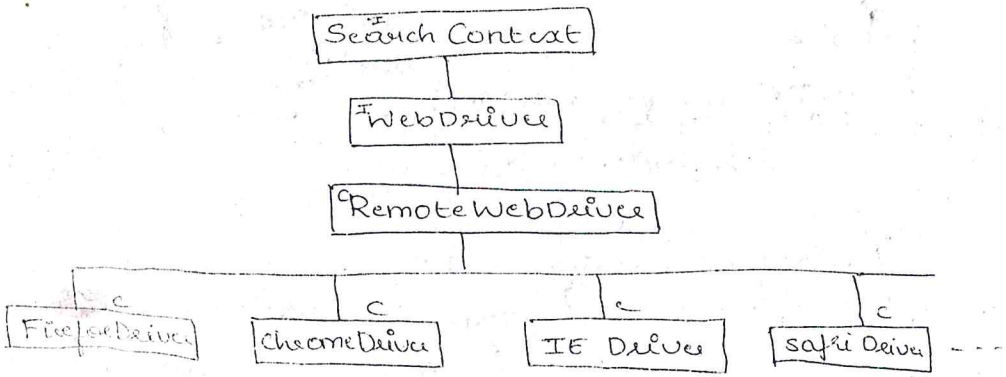
Amazon a = new Sunglass() ← upcasting

For runtime polymorphism,
inheritance
overide
upcasting shld be done

WebDriver



WebDriver



Collected all the .class from the WebDriver & stored in a folder selenium-java-2.47.1.jar.

.jar file → collection of executable file

selenium-java-2.47.1.jar + .jar → browsers

supportable jars

supportable jars

Then Jason collected all these .jar file and stored in selenium-java-2.47.1.zip
23MB

SearchContext is the super class.

NOTE

* WebDriver is a collection of interfaces and class
* WebDriver is a collection of jars
* Technically WebDriver is an interface all browser classes implements WebDriver and SearchContext incomplete methods.
* All WebDriver API's are completely Object oriented.
* WebDriver has native support with all the browsers, so that WebDriver can directly communica -te to browser

* WebDriver is a latest tool available in Selenium community, which removes all the disadvantag of Selenium RC and Selenium IDE

Selenium-2 → WebDriver. Selenium-1 → Selenium IDE

* Selenium 2 supports multiple Browser

IE → InternetExplorerDriver

Firefox → FirefoxDriver

Chrome → ChromeDriver

Safari → SafariDriver

android → AndriodDriver

IOS → IphoneDriver
(iphone OS)

headless browser → HtmlDriver
(Ghost Browser)
cannot see the browser
backend Execution happens
while running scripts

* WebDriver supports multiple language

Java
C#
python
ruby
php
perl

* WebDriver supports multiple platform

Window – OS
Linux – OS
MAC – OS