SDET Foundation Course

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

SELENIUM

“There is No Wrong Time to do Right thing

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

Table of Content

Selenium Session Overview

- Basic Selenium
- TestNG
- Maven
- Advanced Selenium
- ATDD

- Cucumber
- GIT
- Jenkins
- Automation Framework
- API Automation

SELENIUM

History Of Selenium

2004

- Selenium was created by Jason Huggins (engineer at ThoughtWorks).
- He developed a JavaScript library (“JavaScript Task Runner”) that could drive interactions with the page, allowing him to automatically rerun tests against multiple browsers. That library eventually became Selenium Core.
- Challenges due to Same Origin Policy and introduction of Selenium RC by Paul Hammant (another ThoughtWorks engineer).

Selenium 1 = Selenium IDE + Selenium RC + Selenium Grid

2006

- Simon Stewart (engineer at Google) developed WebDriver that spoke directly to the browser using the native method for the browser and Operating system.
- Selenium WebDriver used the concept of native drivers to interact with the Application under test and hence eliminated the need for a proxy server that was used previously in Selenium RC.

2011

- Selenium 1 (i.e. Selenium IDE + Selenium RC + Selenium Grid) merged with Selenium WebDriver to make Selenium more powerful

Selenium 2 = Selenium IDE + (Selenium RC + Selenium WebDriver) + Selenium Grid

- Basically, Selenium and WebDriver merged to form Selenium WebDriver (Selenium 2).

Selenium 2 = Selenium 1 + Selenium WebDriver

2016

- Selenium 3 got release on October, 2016.
- Selenium RC which was internally implementing Selenium Core's JavaScript program libraries is now replaced with the backed WebDriver API implementation. i.e. Whatever the tasks that can be performed only by Selenium RC's JavaScript implementation, can now be performed using the backed WebDriver API implementation which is more flexible

Selenium 3 = Selenium 2 - Selenium RC + Backed WebDriver API for RC

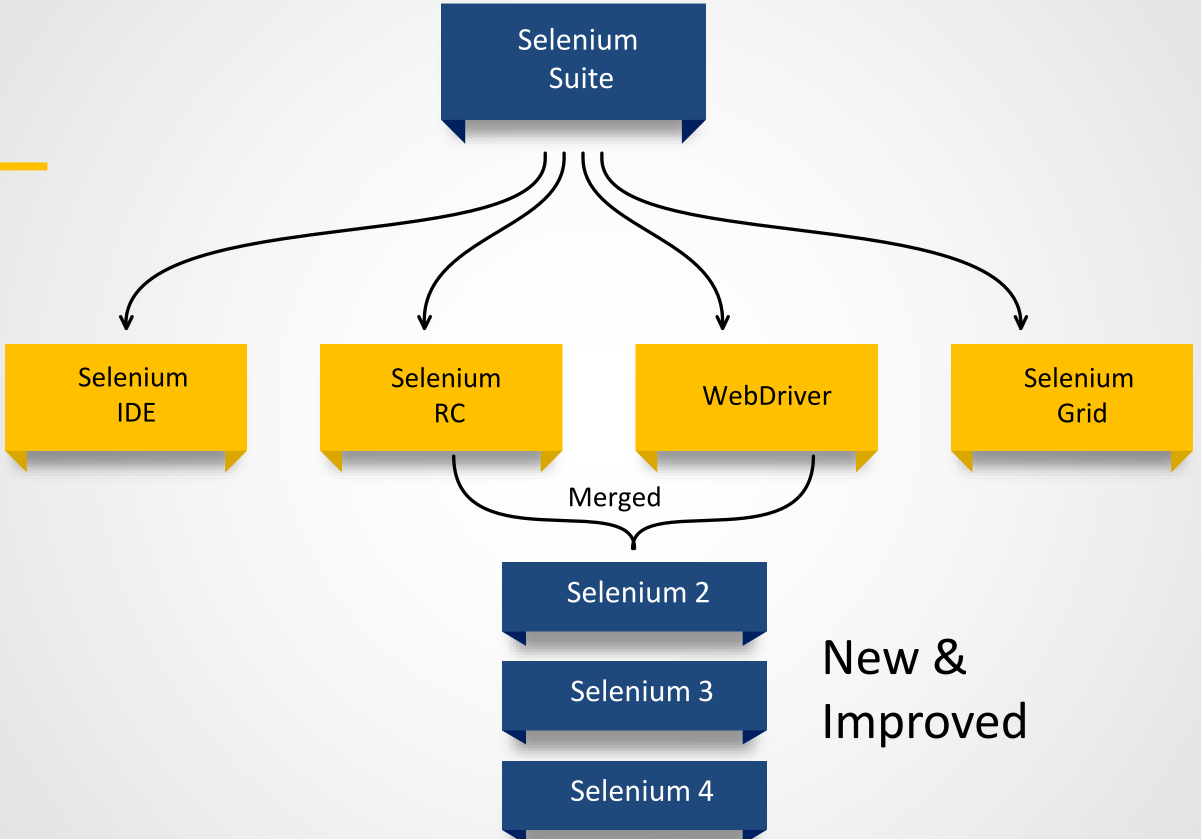
2018

- Selenium team released alpha version Selenium 4 which really got major changes and new features that were announced by Simon Stewart in August 2018 during a conference.
- However, till date only an alpha version is available, and its stable version is yet to be rolled out.

Following are the features of Selenium 4:

- W3C Standardization of WebDriver API
- Improved Selenium Grid
- Introduction of Relative Locators
- Full page Screenshot
- Better Observability and Traceability
- Refreshed and rich documentation

Architectural Overview



New & Improved

Architectural Overview

- Selenium is a set of different tools each with a different approach to support test automation.
  - ❑ Selenium IDE
  - ❑ Selenium RC (Selenium 1)
  - ❑ Selenium WebDriver
  - ❑ Selenium Grid

- Selenium IDE is an extension for Firefox that allows you to record and playback tests.

- Selenium RC provides an API(Application Programming Interface) and library for each of its supported languages such as Java, C#, PHP, Python, etc.

- Selenium Web Driver is a collection of language specific bindings to drive a browser. It is designed in a simpler and more concise programming interface along with addressing some limitations in the Selenium-RC API.

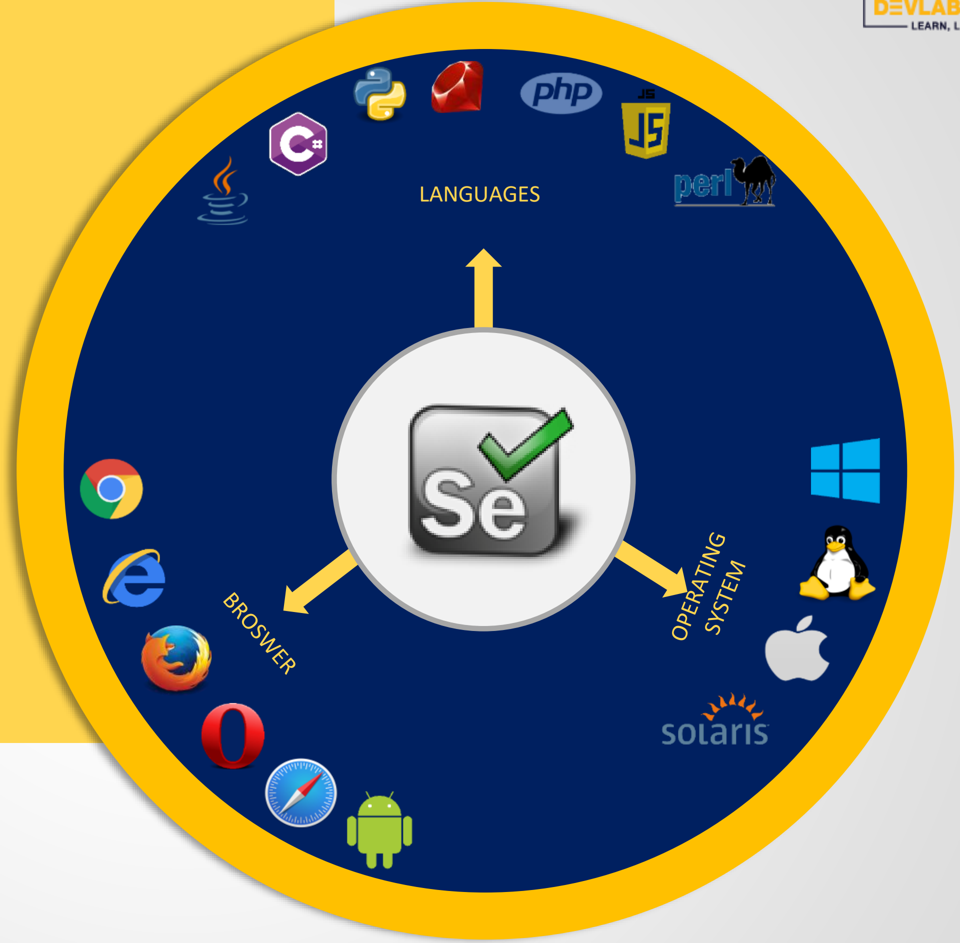
- Selenium Grid is a smart proxy server that makes it easy to run tests in parallel on multiple machines. It allows you to run test cases in different machines across different platforms. It supports distributed test execution. You can also run the test cases in parallel in multiple machines.

Introduction

- ✓ Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms.
- ✓ It supports various languages like Java, PHP, Perl, Python, Ruby, C# and JavaScript.
- ✓ It supports the browser like IE, Mozilla Firefox, Safari, Google Chrome and Opera.
- ✓ It supports the operating systems like Window, Linux, Mac and Solaris.
- ✓ It is very flexible and easy to use as compared to other functional tools, because it supports multiple languages.

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

Introduction



Selenium

Workspace & Project Setup

Selenium – Required Softwares

What all you need to start with Selenium:

collabedit.com

Editors- Eclipse, Intellij, Jboss etc (https://www.eclipse.org/downloads)

Selenium Jars (https://www.seleniumhq.org/download/)

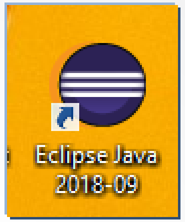
Firefox/Chrome/IE Browsers in your machine

Browser exe – IE, Chrome, Firefox

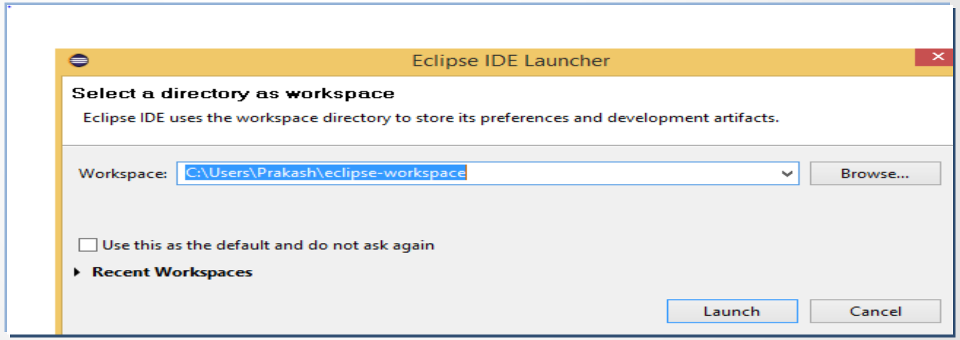
Locators plugin(not Necessary)- Chropath, Firebug, Firepath

Setting-up a project from scratch

Click on this eclipse icon from scratch



Select workspace name,
refer below screen and select launch button-



Eclipse IDE Launcher

Select a directory as workspace

Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\Prakash\eclipse-workspace

Browse...

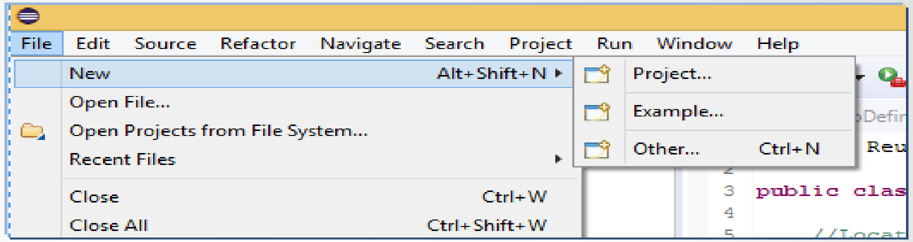
☐ Use this as the default and do not ask again

Recent Workspaces

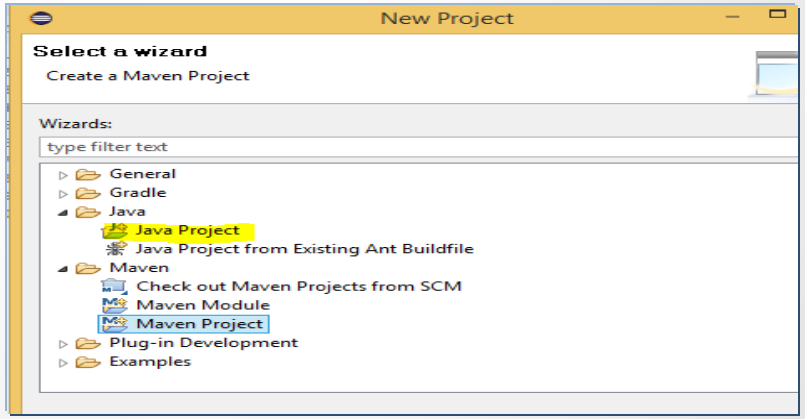
Launch Cancel

Creating a project

Select File menu and click new and select project

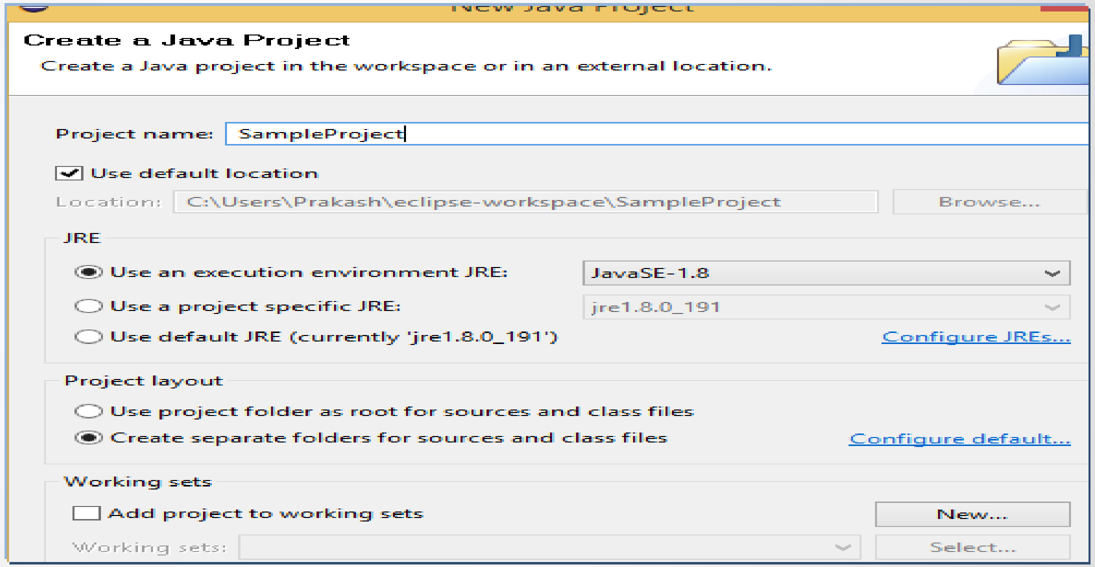


Select Java project



Add Project Details

Enter project name and select JRE details(By default in below screenshot its JavaSE-1.8. Click on Next



Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: SampleProject

[✓] Use default location

Location: C:\Users\Prakash\eclipse-workspace\SampleProject

JRE

(●) Use an execution environment JRE: JavaSE-1.8

( ) Use a project specific JRE: jre1.8.0_191

( ) Use default JRE (currently 'jre1.8.0_191')

Project layout

( ) Use project folder as root for sources and class files

(●) Create separate folders for sources and class files

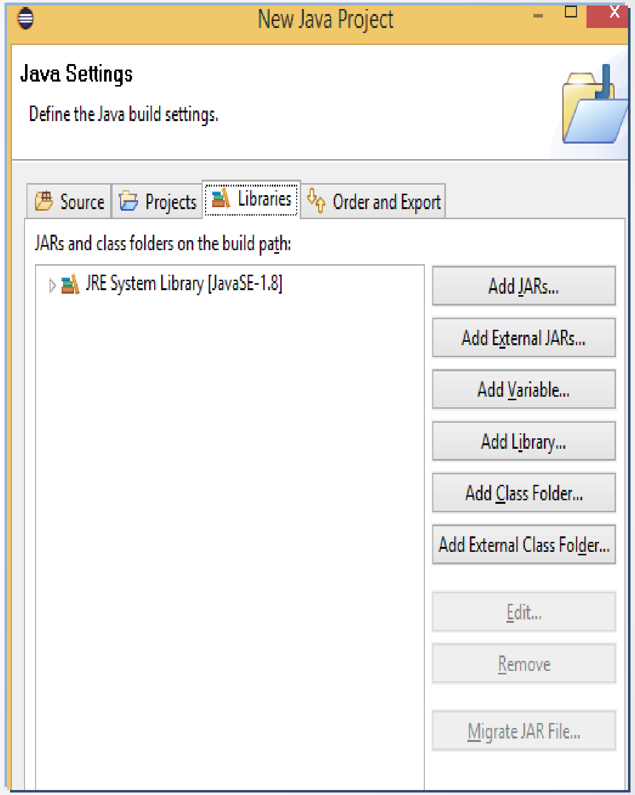
Working sets

[ ] Add project to working sets

Working sets:

Add
Project Details

Select external jar file and select finish


New Java Project

Java Settings

Define the Java build settings.

JARs and class folders on the build path:

JRE System Library [JavaSE-1.8]

Introduction to WebDriver

- ✓ WebDriver is a tool for automation testing of Web Application.
- ✓ WebDriver interacts directly with the browser without any intermediary.
- ✓ It is used to handle multiple frames, multiple browser windows, popups, and alerts.
- ✓ WebDriver is also used for advanced user navigations such as drag-and-drop, mouse hover, etc.

Write your first script in

Selenium

How to launch your browser?

```
public class LaunchChromeBrowser {

    public static void main(String[] args) {

        //Creating a driver object referencing WebDriver interface
        WebDriver driver;

        //Setting the webdriver.chrome.driver property to its executable's location
        System.setProperty("webdriver.chrome.driver",
"/lib/chromeDriver/chromedriver.exe");

        //Instantiating driver object
        driver = new ChromeDriver();

        //Using get() method to open a webpage
        driver.get("https://www.devlabsalliance.com");

        //Closing the browser
        driver.quit();

    }
```

LAB

1

1. Open WebUrl-(www.google.com), maximize the browser using selenium method
2. Launch a webbrowser, open a url in it and close the driver

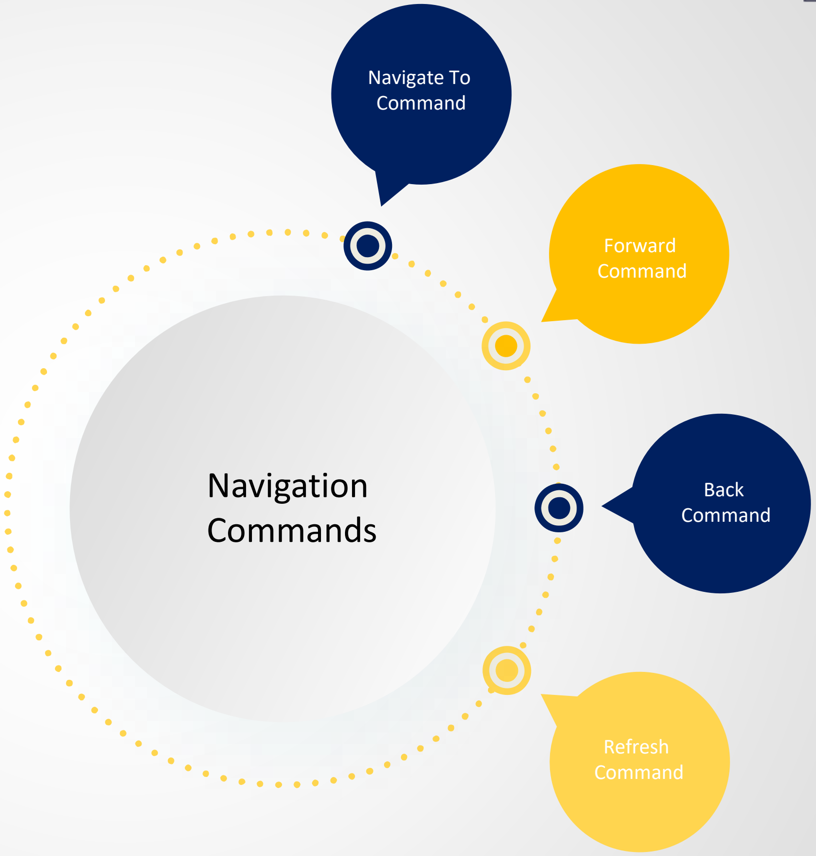
DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

Selenium

Browser & Navigation Commands

WebDriver - Navigation Commands

- Selenium Navigation Commands enables us to open a web page URL, navigate to other page by clicking an element, Back, Forward or Refresh the page using browser's history.
- With the help of Navigating interface, we are able to move back and forward in browser history.



WebDriver- Navigation Commands

Below are the Navigation commands that are present in Selenium WebDriver:

- **navigate().to() :** It is used to open a new browser window and will find the page we provided.

```
driver.navigate().to("https://www.devlabsalliance.com");
```

- **navigate().forward() :** It is used to move the forward page using the Brower's forward command.

```
driver.navigate().forward();
```

- **navigate().back() :** It is used to move to the backward page using the Brower's Back command.

```
driver.navigate().back();
```

- **navigate().refresh() :** It is used to refresh the current page.

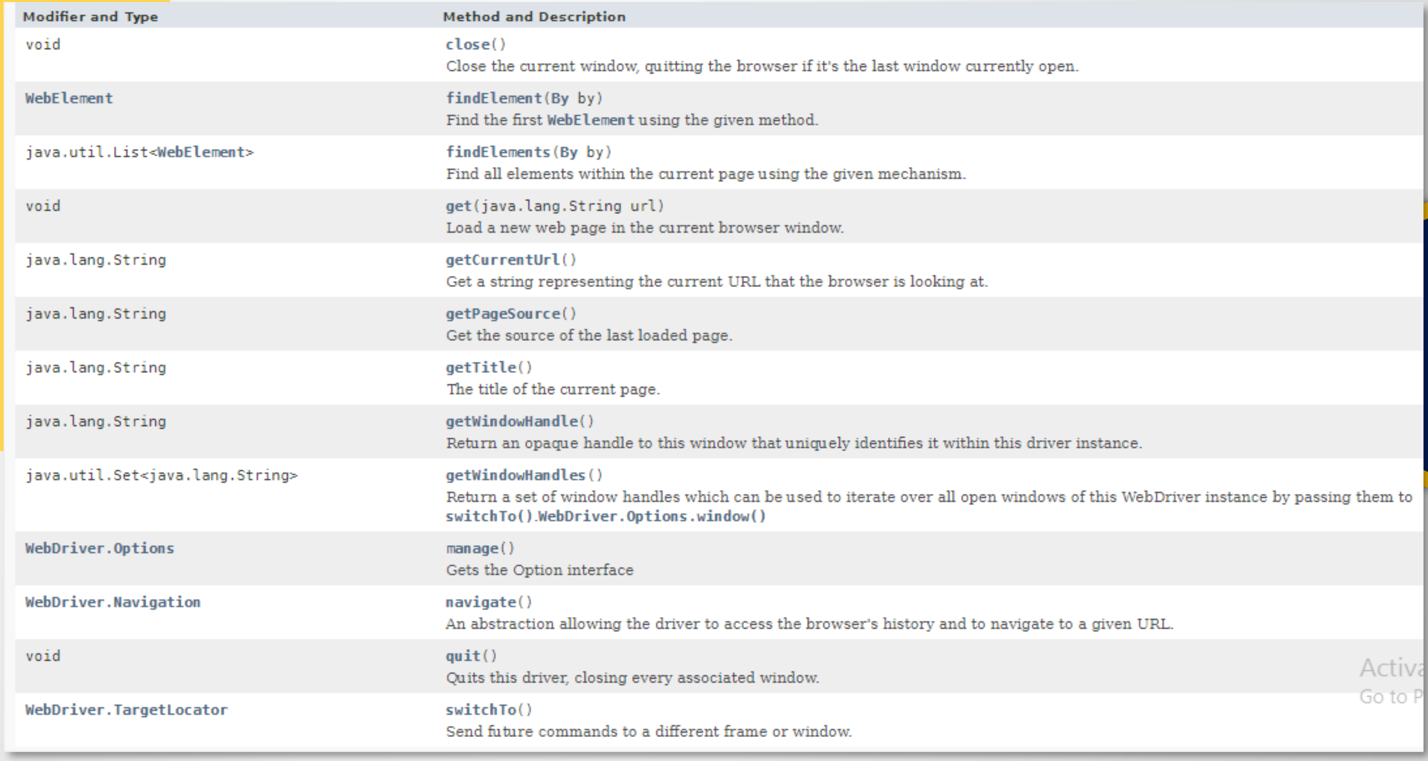
```
driver.navigate().refresh();
```

WebDriver - Navigation Commands

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

public class DlaNavigationClass{

public static void main(String[] args) {
            WebDriver driver ;

            //Set ChromeDriver path
            System.setProperty("webdriver.chrome.driver", "E:\\chromedriver_win32\\chromedriver.exe");

            driver=new ChromeDriver();  //ChromeDriver object creation

            // Using navigate command to navigate to another URL
      driver.navigate().to("https://www.devlabsalliance.com");

      driver.navigate().back();        // Using Back command navigate to back page

      driver.navigate().forward();    // Using Forward command navigate to forward page

      driver.navigate().refresh();    // Using Refresh command Refresh current page
}
}
```

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

WebDriver - Methods

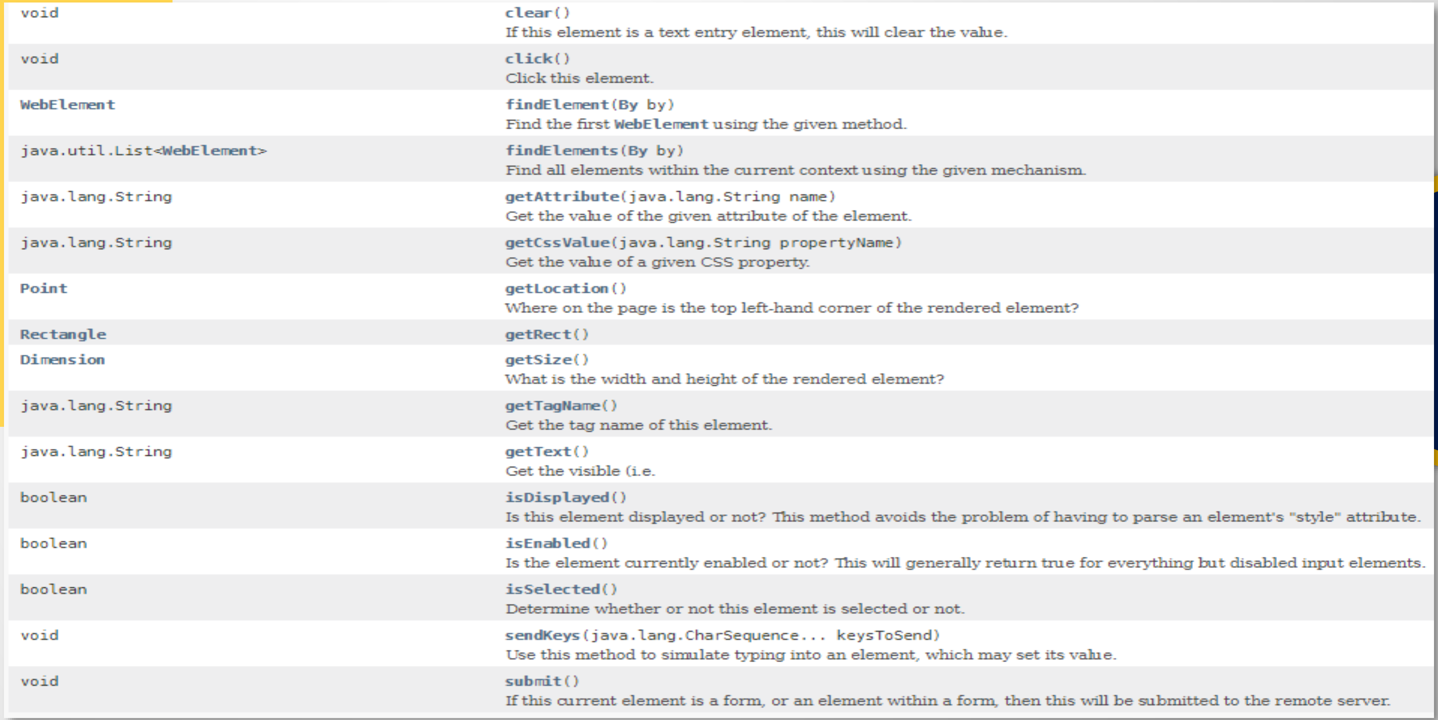


Modifier and Type	Method and Description
void	`close()` Close the current window, quitting the browser if it's the last window currently open.
WebElement	`findElement(By by)` Find the first `WebElement` using the given method.
java.util.List<WebElement>	`findElements(By by)` Find all elements within the current page using the given mechanism.
void	`get(java.lang.String url)` Load a new web page in the current browser window.
java.lang.String	`getCurrentUrl()` Get a string representing the current URL that the browser is looking at.
java.lang.String	`getPageSource()` Get the source of the last loaded page.
java.lang.String	`getTitle()` The title of the current page.
java.lang.String	`getWindowHandle()` Return an opaque handle to this window that uniquely identifies it within this driver instance.
java.util.Set<java.lang.String>	`getWindowHandles()` Return a set of window handles which can be used to iterate over all open windows of this WebDriver instance by passing them to `switchTo().WebDriver.Options.window()`
WebDriver.Options	`manage()` Gets the Option interface
WebDriver.Navigation	`navigate()` An abstraction allowing the driver to access the browser's history and to navigate to a given URL.
void	`quit()` Quits this driver, closing every associated window.
WebDriver.TargetLocator	`switchTo()` Send future commands to a different frame or window.

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

WebDriver - Methods

WebElement API (I): Represents an HTML element. Generally, all interesting operations to do with interacting with a page will be performed through this interface.



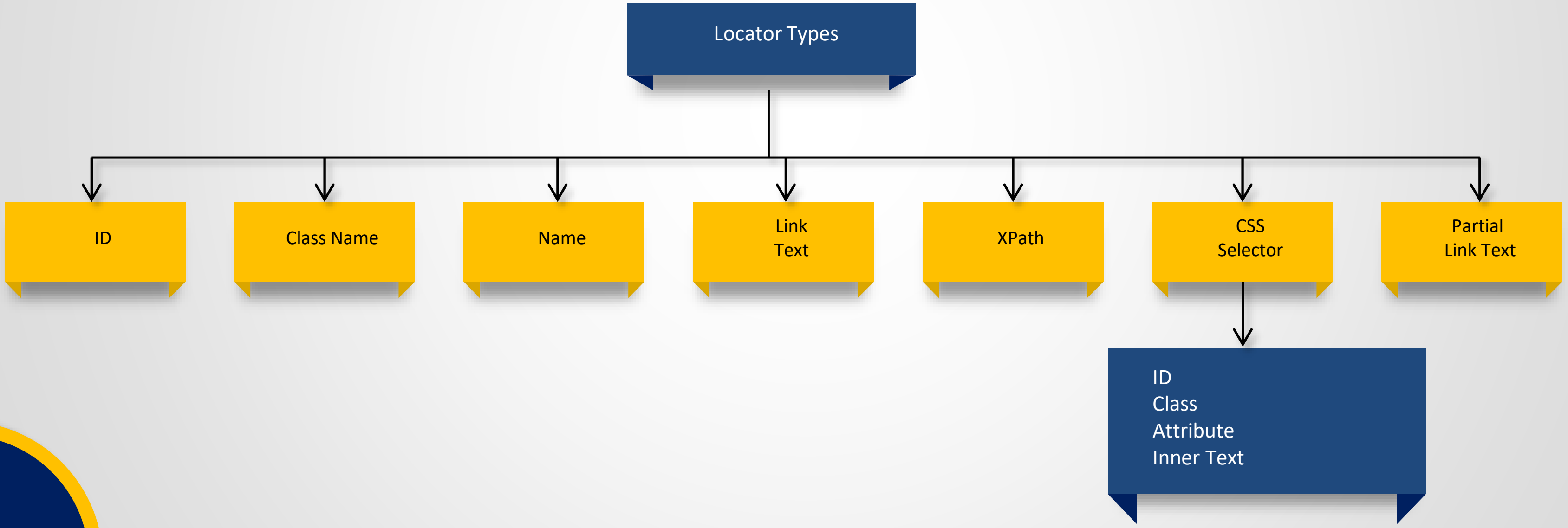
void	**clear**() If this element is a text entry element, this will clear the value.
void	**click**() Click this element.
WebElement	**findElement**(By by) Find the first WebElement using the given method.
java.util.List<WebElement>	**findElements**(By by) Find all elements within the current context using the given mechanism.
java.lang.String	**getAttribute**(java.lang.String name) Get the value of the given attribute of the element.
java.lang.String	**getCssValue**(java.lang.String propertyName) Get the value of a given CSS property.
Point	**getLocation**() Where on the page is the top left-hand corner of the rendered element?
Rectangle	**getRect**()
Dimension	**getSize**() What is the width and height of the rendered element?
java.lang.String	**getTagName**() Get the tag name of this element.
java.lang.String	**getText**() Get the visible (i.e.
boolean	**isDisplayed**() Is this element displayed or not? This method avoids the problem of having to parse an element's "style" attribute.
boolean	**isEnabled**() Is the element currently enabled or not? This will generally return true for everything but disabled input elements.
boolean	**isSelected**() Determine whether or not this element is selected or not.
void	**sendKeys**(java.lang.CharSequence... keysToSend) Use this method to simulate typing into an element, which may set its value.
void	**submit**() If this current element is a form, or an element within a form, then this will be submitted to the remote server.

LAB 2

1. Write a script which prints page title of facebook.com login page in Chrome Browser
2. Write a script to print page url of google.com
3. Write a script to print page source of google.com

Locators in Selenium

- Locator is a command that is used to locate the Web Elements like Text Box, Buttons, CheckBoxes, etc. and is used for further operations.
- It is one of the essential components of Selenium that helps scripts to uniquely identify the WebElements.



DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

Selenium

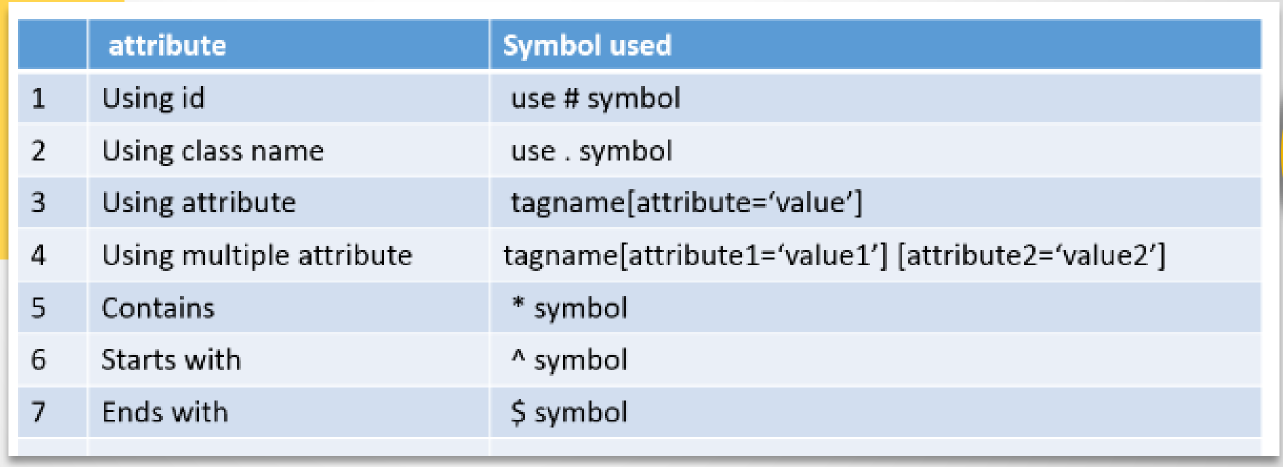
CCS Selector

CssSelector-

How to Use

Selenium supports multiple locators which will help you to identify web element and perform the operation based on your requirements. We will discuss about CSS Selector in Selenium and you will feel the importance of the same

Symbol used while writing CSS selector in Selenium Webdriver:



	attribute	Symbol used
1	Using id	use # symbol
2	Using class name	use . symbol
3	Using attribute	tagname[attribute='value']
4	Using multiple attribute	tagname[attribute1='value1'] [attribute2='value2']
5	Contains	* symbol
6	Starts with	^ symbol
7	Ends with	$ symbol

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

Selenium

Xpath

Locators - Xpaths

- Xpath stands for XMLPath Language where XML is extensible Markup Language.
- Xpath uses “path like” syntax to identify and navigate to nodes in an XML document.
- Xpath contains over 200 built-in functions.
- XPath is a technique to traverse through the HTML structure of a webpage. It helps in navigating through the XML structure.
- XPath expressions are capable of locating web elements on the web page that are complex and changes dynamically. Dynamic web elements are those whose attributes change dynamically when the web page is refreshed or some dynamic operation is performed on it.
- XPath is basically used when Selenium cannot the locate the web elements with the help of usual locators like id, name, class etc.

Locators - Xpaths

Sample XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book>
<title lang="en">Harry Potter</title>
<price>29.99</price>
</book>

<book>
<title lang="en">Learning XML</title>
<price>39.99</price>
</book>
</bookstore>
```

Xpath –
How to Use

Syntax for XPath:

XPath contains the path of the element situated at the web page.
Standard syntax for creating XPath is:

```
Xpath=//tagname[@attribute='value']
```

- // : Select current node.
- Tagname: Tagname of the particular node.
- @: Select attribute.
- Attribute: Attribute name of the node.
- Value: Value of the attribute.

Xpath – Types

Types of X-path

There are two types of XPath: **1) Absolute Xpath 2) Relative Xpath**

Absolute XPath:

- It is the direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed.
- The key characteristic of XPath is that it begins with the single forward slash(/) ,which means you can select the element from the root node. Below is the example of an absolute xpath expression of the element shown in the below screen.
- **Absolute xpath:** html/body/div[1]/section/div[1]/div/div/div/div[1]/div/div/div/div/div[3]/div[1]/div/h4[1]/b

Relative xpath: For Relative Xpath the path starts from the middle of the HTML DOM structure. It starts with the double forward slash (//)

Xpath functions

Relationship of Nodes:

- Parent
- Children
- Siblings
- Ancestors
- Descendants

Useful functions:

- last()
- position()
- text()
- not()
- and
- or
- String-length()

LAB 3

1. Write Xpath for search box in Google search page
2. Write CssSelector for search box in Google search page
3. Write CssSelector for Facebook login page- All elements- Username, password and button

WebDriver Methods

Writing First selenium Test-

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class TestClass
{

public static void main(String[] args) {
// TODO Auto-generated method stub
            WebDriver driver ;
            // TODO Auto-generated method stub D:\chromedriver_win32
            String exePath = "E:\\chromedriver_win32\\chromedriver.exe";
            System.setProperty("webdriver.chrome.driver", exePath);
            //ChromeDriver object creation
            driver=new ChromeDriver();
            //Maximizing  window
            driver.manage().window().maximize();
            driver.get("http://www.google.com");
            driver.findElement(By.name("q")).sendKeys("DevLabsalliance.com");
}

}
```

Understanding the script

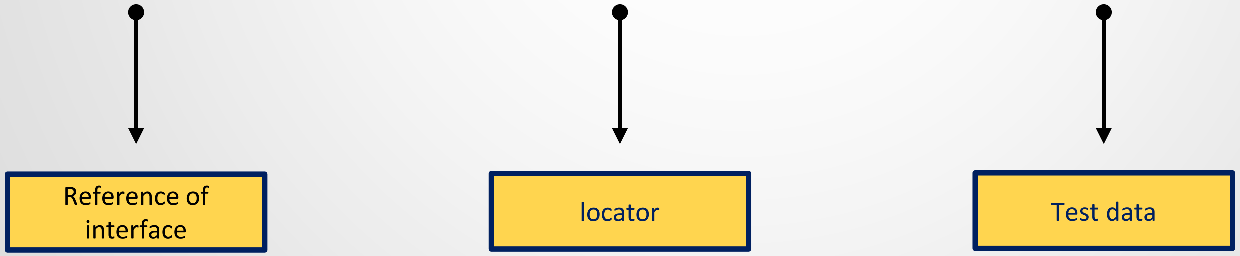
Lets understand what we have scripted in previous page. Lets understand this way-
Every web-driver interaction with web has 3 different components-

1- Driver – Reference of Interface

2- Locator

3-Test data

```
driver.findElement(By.name("q")).sendKeys("DevLabsalliance.com")
```



Selecting CheckBox & Radio button

- Radio Button and Checkbox can be selected or de-selected by using click() method.
- First we need to locate the Radio button or Checkbox WebElement using any Locator and then will call the click() method on that WebElement.

Syntax :

```
WebElement radioBtn = driver.findElement(By.id(“rad_btn”));
radioBtn.click();

WebElement chkBox = driver.findElement(By.id(“chk_box”));
chkBox.click();
```

Select Class

- The Select Class in Selenium is a method used to implement the HTML SELECT tag.
- Html select tag provides helper methods to select and deselect the elements. The Select class is an ordinary class so New keyword is used to create its object and it specifies the web element location.
- The 'Select' class in Selenium WebDriver is used for selecting and deselecting option in a dropdown and also used to handle Multiple Select operations.

Syntax:

```
WebElement dropDown = driver.findElement(By.id(“drop_down"));
Select testDropdown = new Select(dropDown);
```

Select Method

- selectByVisibleText & deselectByVisibleText:
  This method is used to select one of the options in a drop-down or an option in multiple selection boxes on the basis of text over options.
  Syntax : dropdown.selectByVisibleText("SDET Training");

- selectByIndex & deselectByIndex :
  This method used to select an option based on its index, beginning with 0.
  Syntax : dropdown. selectByIndex(2);

- selectByValue & deselectByValue:
  This method used to select an option based on its 'value' attribute.
  Syntax : dropdown.selectByValue("DevLabs");

- deselectAll:
  This method used to clear all selected items.
  Syntax : dropdown.selectByValue("DevLabs");

Selenium

Implicit & Explicit wait

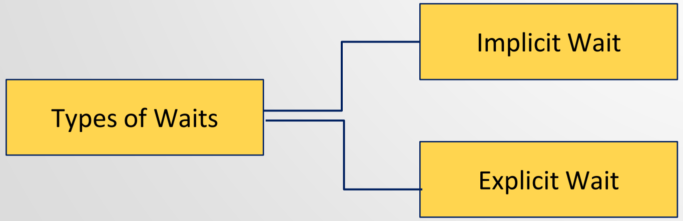
Waits in Selenium

- Waits are used in test automation to handle application delay in loading page/web objects.

Why do we need Waits in Selenium?

Most of the web applications are developed using Ajax and Javascript. When a page is loaded by the browser, the elements may load at different time intervals.

So, we would not be able to identify the element and when element is not located, it will throw an “ElementNotVisibleException” exception. So, to avoid this, we use Waits in Selenium.



Waits - Implicit

Implicit Wait:

- The Implicit Wait will tell the web driver to wait for certain amount of time before it throws a "No Such Element Exception".
- The default setting is 0.
- Once we set the time, web driver will wait for that time before throwing an exception.

Syntax : driver.manage().timeouts().implicitlyWait(TimeOut, TimeUnit.SECONDS);

Waits - Explicit

Explicit Wait-

- The explicit wait is used to tell the Web Driver to wait for certain conditions (Expected Conditions) or the maximum time exceeded before throwing an "ElementNotVisibleException" exception.
- The explicit wait is an intelligent kind of wait, but it can be applied only for specified elements. Explicit wait gives better options than that of an implicit wait as it will wait for dynamically loaded Ajax elements.

```
WebDriverWait wait = new WebDriverWait(WebDriverRefrence,TimeOut);

WebDriverWait wait=new WebDriverWait(driver, 20);

WebElement webelementlink;

webelementlink = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(
"/html/body/div[1]/section/div[2]/div/div[1]/div/div[1]/div/div/div/div[2]/div[2]/div/div/div/div/div[1]
/div/div/a/i")));
```

Waits - Explicit

The following are the Expected Conditions that can be used in Explicit Wait

- alertIsPresent()
- elementSelectionStateToBe()
- elementToBeClickable()
- elementToBeSelected()
- frameToBeAvaliableAndSwitchToIt()
- invisibilityOfTheElementLocated()
- invisibilityOfElementWithText()
- presenceOfAllElementsLocatedBy()
- presenceOfElementLocated()
- textToBePresentInElement()
- textToBePresentInElementLocated()
- textToBePresentInElementValue()
- titleIs()
- titleContains()
- visibilityOf()
- visibilityOfAllElements()
- visibilityOfAllElementsLocatedBy()
- visibilityOfElementLocated()

Waits - Custom

Another way of implementing waits is using below custom waits-

```
public static WebElement isElementPresent(WebDriver driver,String xpath,int time){
WebElement ele = null;

for(int i=0;i<time;i++)
{
try{
ele=driver.findElement(By.xpath(xpath));
break;
}
catch(Exception e)
{
try
{
Thread.sleep(1000);
} catch (InterruptedException e1)
{
System.out.println("Waiting for element to appear on DOM");
}
}
}
return ele;
}
```

Waits in Selenium

FluentWait Command

FluentWait instance defines the maximum amount of time to wait for any condition, as well as frequency with which to check the condition before throwing an “ElementNotVisibleException” exception.

Syntax:

```
Wait wait = new FluentWait(WebDriver reference)
.withTimeout(Duration.ofSeconds(SECONDS)) .pollingEvery(Duration.ofSeconds(SECONDS))
.ignoring(Exception.class);
```

PageLoadTimeOut Command

It sets the amount of time to wait for a page-load to complete before throwing an error. If the timeout is negative, page loads can be indefinite.

Syntax:

```
driver.manage().timeouts().pageLoadTimeout(100, SECONDS);
```

Waits in Selenium

SetScriptTimeout Command

It sets the amount of time to wait for an asynchronous script to finish execution before throwing an error. If the timeout is negative, then the script will be allowed to run indefinitely.

Syntax:

```
driver.manage().timeouts().setScriptTimeout(100,SECONDS);
```

Sleep Command

It forces the browser to wait for a specific time.

Syntax:

```
thread.sleep(1000);
```

Selenium

Alerts & Windows Handling

Alerts in Selenium

- An Alert is a small message box that appears on the screen to give some information or notification to the users.
- It notifies the user with some specific information, or will ask for some permissions to perform certain task and it provides warning messages as well.

Types of Alerts

1. Simple Alert: It simply displays some information or warning on the screen.
2. Prompt Alert: It will ask user to provide some input into it.
3. Confirmation Alert: It asks user for permission to perform certain task.

Methods in Alert

An Alert interface provides the below methods that are widely used in Selenium WebDriver:

- void dismiss(): It is used to click on the "Cancel" button of the alert.
  Syntax: driver.switchTo().alert().dismiss();

- void accept(): It is used to click on "OK" button of the alert.
  Syntax: driver.switchTo().alert().accept();

- String getText(): It is used to capture the alert message.
  Syntax: driver.switchTo().alert().getText();

- void sendKeys(): It is used to send some data to alert box.
  Syntax: driver.switchTo().alert().sendKeys("DevLabs");

Handling Selenium Window

- When we have multiple windows in any web application, sometimes it is required to switch control among several windows from one another in order to do the automation.
- After completion of the operation in another window, it has to return to main window.

To handle multiple windows, selenium web driver use below 2 methods:

- driver.getWindowHandles();

  It is used to handle all windows that are opened by web driver, we use "Driver.getWindowHandles()" to switch from one window to another in a web application. Its return type is Iterator<String> or Set<String>.

- driver.getWindowHandle();

  When any web application gets opened, we need to handle the main window by ”driver.getWindowHandle()”. This will handle the current window that uniquely identifies it within this driver instance. Its return type is String.

LAB 4

1. Create a script to handle:
   - Simple Alert
   - Confirmation Alert
   - Prompt Alert
2. Write a script to handle multiple windows in application

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

Action Class

Action Class

- Action class is a built-in feature provided by Selenium for handling Keyboard and Mouse events.
- Handling these events in Selenium WebDriver includes operations such as drag and drop, clicking on multiple elements with the control key, etc. These operations are performed using the advanced user interactions API in Selenium Webdriver.

Action class is defined and invoked using the following syntax:

```
Actions action = new Actions(driver);
action.moveToElement(element).click().perform();
```

Action Class

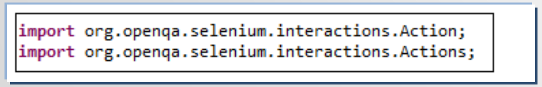
Handling Keyboard & Mouse Events-

Handling special keyboard and mouse events are done using the Advanced User Interactions API. It contains the Actions and the Action classes that are needed when executing these events.

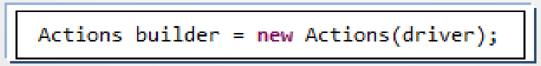
Step 1-
Import Action and Actions classes

Step 2-
Instantiate a new Actions object

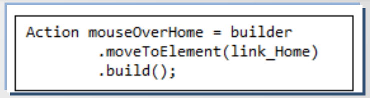
Step 3-
Instantiate an Action using Actions object in Step 2



```
import org.openqa.selenium.interactions.Action;
import org.openqa.selenium.interactions.Actions;
```



```
Actions builder = new Actions(driver);
```



```
Action mouseOverHome = builder
        .moveToElement(link_Home)
        .build();
```

Methods of Action Class

Selenium provides the following methods to handle Mouse and Keyboard actions:

Mouse Actions:

- doubleClick(): It is used to perform double click on the element.
- clickAndHold(): It is used to perform long click on the mouse without releasing it.
- moveToElement(): It is used to shift the mouse pointer to the center of the element.
- dragAndDrop(): It is used to drag the element from one point and drop to another.
- contextClick(): It is used to perform right-click on the mouse.

Keyboard Action:

- sendKeys(): It is used to send a series of keys to the element.
- keyUp(): It is used to perform key release.
- keyDown(): It is used to perform a keypress without release.

LAB 5

1. Write a script to handle various Mouse events.

2. Write a script to handle various Keyboard events.

iFrames

in Selenium

iFrames

- iFrame in Selenium Webdriver is a web page or an inline frame that is embedded in another web page or an HTML document embedded inside another HTML document.
- iFrame is often used to add content from other sources like advertisement into a webpage.
- iFrame is defined with the **<iframe>** tag.

To Switch between iFrames we have to use the driver's **switchTo().frame** command. We can use the switchTo().frame() in three ways:

- By index
- By Name or Id
- By Web Element

iFrames

- Switch to Frame by Index:
  Index is one of the attributes to switch through the iFrame.
  Syntax: driver.switchTo().frame(0);

- Switch to Frame by Name or Id:
  Another attribute through which we can switch to the iFrame is by its Name or Id.
  Syntax: driver.switchTo().frame("iFrame1");
  driver.switchTo().frame("id of the element");

- Switch to Frame by Web Element:
  We can switch to iFrame using Web Element as well.
  Syntax: driver.switchTo().frame(WebElement);

- Switching back to Main Frame:
  Once we are done with working on iFrame, we need to return to the parent frame.
  Syntax: driver.switchTo().defaultContent();

JavaScript Executor

JavaScript Executor is an Interface that helps to execute JavaScript through Selenium Webdriver. JavaScript Executor provides two methods "executescript" & "executeAsyncScript" to run JavaScript on the selected window or current page.

Syntax:

```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript(Script,Arguments);
```

- Script – This is the JavaScript that needs to execute.
- Arguments – It is the arguments to the script. It's optional.

JavaScript Executor

JavaScript Executor is an Interface that helps to execute JavaScript through Selenium Webdriver. JavaScript Executor provides two methods "executescript" & "executeAsyncScript" to run JavaScript on the selected window or current page.

Syntax:

```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript(Script,Arguments);
```

Script – This is the JavaScript that needs to execute.
Arguments – It is the arguments to the script. It's optional.

JavaScript Executor

Generate Alert Pop Window:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
Js.executeScript("alert(‘DevLabs Alliance');");
```

Click Action:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
Js.executeScript("arguments[0].click();", element);
```

Refresh Browser:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
Js.executeScript("history.go(0)");
```

Get Title of WebPage:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
string sText =  js.executeScript("return document.title;").toString();
```

Scroll Page:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
  //Vertical scroll - down by 150  pixels
  js.executeScript("window.scrollBy(0,150)")
```

Exception in Selenium..1/2

Any unwanted occurrence in application functionality which impacts its normal behaviour is exception and desired output is not achieved.

- Checked Exception: Checked exception is handled during compile time and it gives the compilation error if it is not caught and handled during compile time.
  Example**:** FileNotFoundException, IOException etc.

- Unchecked Exception: In case of the unchecked exception, a compiler does not mandate to handle. The compiler ignores during compile time.

Common Selenium exception-

- NoSuchElementException
- NoSuchWindowException
- NoSuchFrameException
- NoAlertPresentException
- InvalidSelectorException
- ElementNotVisibleException
- ElementNotSelectableException
- TimeoutException
- NoSuchSessionException
- StaleElementReferenceException

Exception in Selenium..1/2

- ElementNotVisibleException: If selenium tries to find an element but the element is not visible within the page.
- NoAlertPresentException: If a user tries to handle an alert box but the alert is not present.
- NoSuchAttributeException: While trying to get attribute value but the attribute is not available in DOM.
- NoSuchElementException: This exception is due to accessing an element which is not available on the page.
- WebDriverException: Exception comes when a code is unable to initialize WebDriver.

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

Selenium Grid

What is Selenium Grid?

- Selenium Grid is a feature in Selenium that allows you to run test cases in different machines across different platforms. The control of triggering the test cases is on the local machine, and when the test cases are triggered, they are automatically executed by the remote machine. Suppose you have 5 test cases. Your local machine is running multiple applications, so you want to run your test cases in a remote machine. You need to configure the remote server so that the test cases can be executed there.

- It supports distributed test execution. Initially, you have a local machine where you write the test cases and executes on the same machine. However, in a big organization, you have multiple test cases, and it's not possible to run all the test cases in the same machine. In a large organization, you have multiple servers, so local machine distributes the test cases across different machines/servers.

- You can also run the test cases in parallel in multiple machines on Selenium Grid.

Why Selenium Grid?

Selenium Grid is useful because of the following reasons:

- Run on different platforms: We can run the test cases in different platforms, so it is a platform independent. For example, you have a hub which has internet explorer 9.
  Hub has internet explorer 9 as many older websites support IE 9. Now we want to run the test cases in different browser such as Internet Explorer. As we know that only one version of a browser can be installed on a computer. You need to configure the node to which you want to send your test cases.

- Parallel execution: If we set up the Selenium Grid, we can run the multiple cases at the same time. This saves time in running the test suites.

How to take Screenshot?

A screenshot in Selenium is used for bug analysis. Users need to use the TakeScreenshot method that notifies the WebDriver to take the screenshot and store it.

Syntax:

```
File file = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
String screenshotBase64 =
((TakesScreenshot)driver).getScreenshotAs(OutputType.BASE64);
```

*OutputType defines the output type for the required screenshot.

What is Cross Browser Testing?

Cross Browser Testing is a type of functional test to check that your web application works as expected in different browsers.

A web application can be opened in any browser by the end user. For example, some people prefer to open any site in Firefox browser, while other's can be using Chrome browser or IE. So, we need to ensure that the web application will work as expected in all popular browsers so that more people can access it and use it.

Cross Browser Testing
Se

Launching Different Browsers

Launching Firefox Browser:
To launch Firefox browser, we need to download Gecko driver that will interact with the Firefox browser.

Syntax:

```
// Defining System Property for the FirefoxDriver
System.setProperty("webdriver.gecko.driver ", "D:\\geckodriver.exe");
// Instantiate a Firefox driver.
WebDriver driver=new FirefoxDriver();
```

Launching Internet Explorer:
To launch Internet Explorer, we need to download InternetExplorerDriver.

Syntax:

```
// Defining System Property for the IEDriver
System.setProperty("webdriver.ie.driver", "D:IE
DriverServerIEDriverServer.exe");
// Instantiate a Iedriver class
WebDriver driver=new InternetExplorerDriver();
```

Selenium

TestNG

TestNG

TestNG is an automation testing framework:
-NG stands for "Next Generation"
-TestNG is inspired from Junit

Following are key features of TestNG:

- Generate the report in a proper format including passed, failed, skipped test cases separately.
- Multiple test cases can be grouped more easily by converting them into testng.xml file.
- Prioritize which test case should be executed first.
- The same test case can be executed multiple times without loops just by using keyword called 'invocation count.'
- Using testng, you can execute multiple test cases on multiple browsers, i.e., cross browser testing.
- The testing framework can be easily integrated with tools like Maven, Jenkins, etc.
- Annotations used in the testing are very easy to understand ex: @BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest
- Uncaught exceptions are automatically handled by TestNG without terminating the test prematurely.

TestNG

Below are the major Topics/features of TestNG:

- Annotations with Order of Execution of Annotation
- Setting Priority
- Ignore Test case
- Thread pool Size
- Running Test case multiple times
- Create Groups
- Parameter passing (XML)
- Parameter passing using Data Provider
- Handle Exceptions
- Method dependency
- Group dependency
- Timeout
- Soft Assertion
- Parallel Execution
- Listeners

TestNG - Configuration

1)Launch the Eclipse IDE and from Help menu, click “Install New Software”.

2) You will see a dialog window, click “Add” button.

3) Type name as you wish, lets take “TestNG” and type “http://beust.com/eclipse/” as location. Click OK.

4) You come back to the previous window but this time you must see TestNG option in the available software list. Just Click TestNG and press “Next” button.

5) Click “I accept the terms of the license agreement” then click Finish.

6) You may or may not encounter a Security warning, if in case you do just click OK.

7) Click Next again on the succeeding dialog box until it prompts you to Restart the Eclipse.

8) You are all done now, just Click Yes.

TestNG - Annotation -1

Sr.No.	Test NG- Annotation & Description
1	@BeforeSuite The annotated method will run only once before all tests in this suite have run.
2	@AfterSuite The annotated method will run only once after all tests in this suite have run.
3	@BeforeClass The annotated method will run only once before the first test method in the current class is invoked.
4	@AfterClass The annotated method will run only once after all the test methods in the current class have run.
5	@BeforeTest The annotated method will run before any test method belonging to the classes inside the <test> tag is run.
6	@AfterTest The annotated method will run after all the test methods belonging to the classes inside the <test> tag have run.
7	@BeforeGroups The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.

TestNG - Annotation - 2

Sr.No.	TestNG-Annotation & Description
7	@BeforeGroups The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.
8	@AfterGroups The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.
9	@BeforeMethod The annotated method will run before each test method.
10	@AfterMethod The annotated method will run after each test method.
11	@Factory Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[ ].
12	@Listeners Defines listeners on a test class.
13	@Parameters Describes how to pass parameters to a @Test method.
14	@Test Marks a class or a method as a part of the test.

TestNG - XML – Key Definitions

- Thread-count: This is used for parallel execution, based on the number mentioned in the test script. It will execute in parallel or sequential order.

- Verbose: It is used to log the execution details in the console. The value should be 1-10. The log details in the console window will get more detailed and clearer as you increase the value of the verbose attribute in the testng.xml configuration file.

- Name: Name of the suite. Here it is “Test Suite“

- Parallel: To run scripts parallel, value can be tests/classes/methods/suites. Default value is none

TestNG - Multiple class file execution

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-
1.0.dtd"> <suite thread-count="1" verbose="1"
name="Gmail Suite" annotations="JDK"
parallel="tests"> <test name="Test1">
                <classes>
                      <class name="ClassName1"/>
                    </classes>
           </test>
           <test name="Test2">
                      <classes> <class name="Classname2"/>
           </classes> </test> </suite>
```

Listeners: Definition

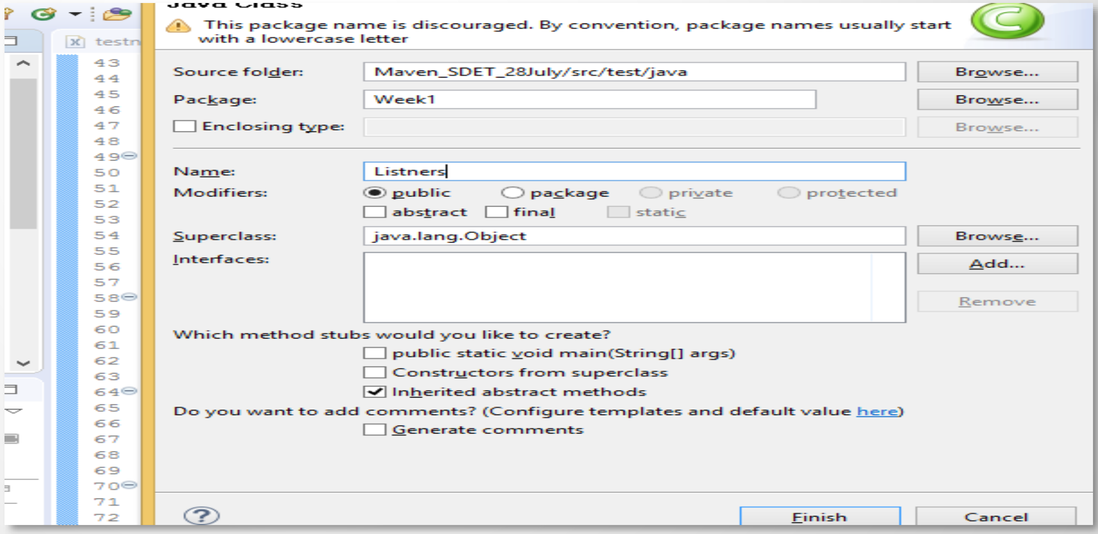
There are several interfaces that allow you to modify TestNG's behaviour. These interfaces are broadly called "TestNG Listeners". Here are a few listeners:

- IAnnotationTransformer
- IAnnotationTransformer2
- IHookable
- IInvokedMethodListener
- IMethodInterceptor
- IReporter
- ISuiteListener
- ITestListener

Listeners: Implementation – 1/4

There are several interfaces that allow you to modify TestNG's behavior. These interfaces are broadly called "TestNG Listeners". Here are a few listeners:

Step 1- Create a simple class and implement ITestListener listener



This package name is discouraged. By convention, package names usually start with a lowercase letter

Source folder: Maven_SDET_28July/src/test/java

Package: Week1

Enclosing type: ☐

Name: Listners

Modifiers: [✓ public] package private protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

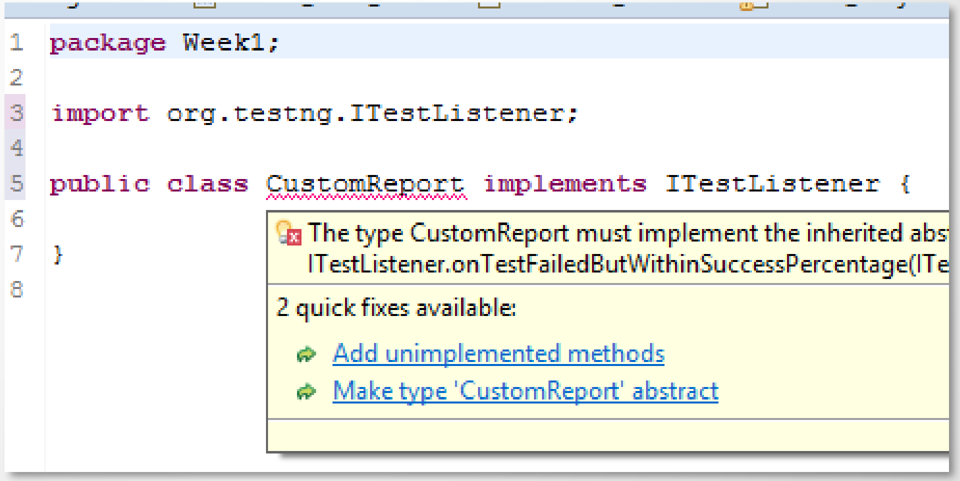
Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Listeners: Implementation – 2/4

There are several interfaces that allow you to modify TestNG's behavior. These interfaces are broadly called "TestNG Listeners".

Step 2- Override all unimplemented methods(By Clicking Add unimplemented methods



```
package Week1;

import org.testng.ITestListener;

public class CustomReport implements ITestListener {

}
```

The type CustomReport must implement the inherited abs
ITestListener.onTestFailedButWithinSuccessPercentage(ITe

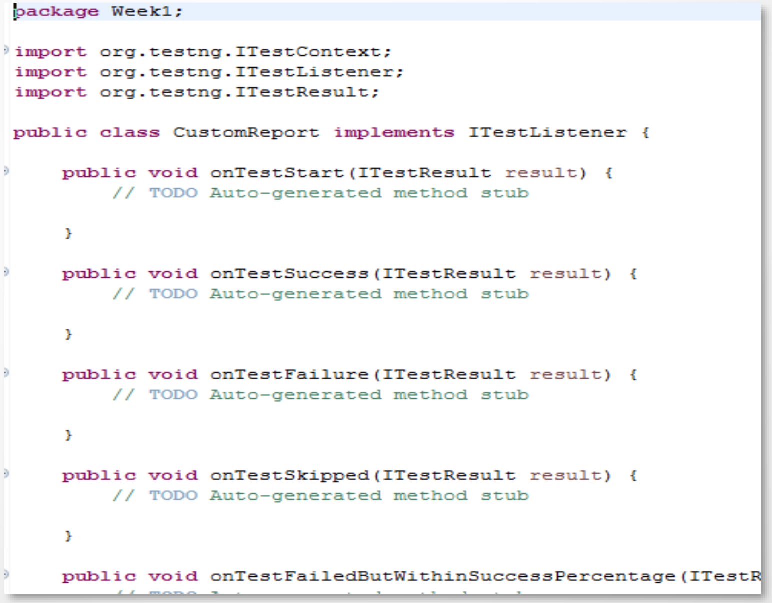
2 quick fixes available:

- Add unimplemented methods
- Make type 'CustomReport' abstract

Listeners: Implementation – 3/4

Once after implementing all unimplemented methods, below methods will be created.

As per need, below method's can be override

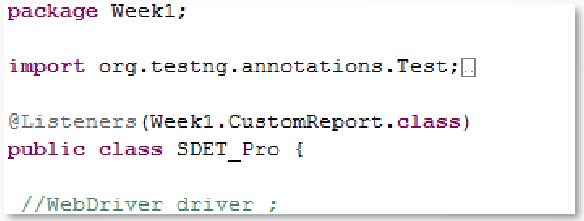


```
package Week1;

import org.testng.ITestContext;
import org.testng.ITestListener;
import org.testng.ITestResult;

public class CustomReport implements ITestListener {

    public void onTestStart(ITestResult result) {
        // TODO Auto-generated method stub

    }

    public void onTestSuccess(ITestResult result) {
        // TODO Auto-generated method stub

    }

    public void onTestFailure(ITestResult result) {
        // TODO Auto-generated method stub

    }

    public void onTestSkipped(ITestResult result) {
        // TODO Auto-generated method stub

    }

    public void onTestFailedButWithinSuccessPercentage(ITestR
```

Listeners: Implementation – 4/4

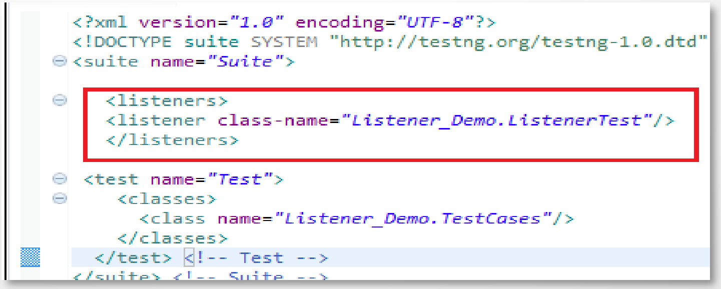
Implement this listener in our regular project class i.e. "TestCases". There are two different ways to connect to the class and interface.

- The first way is to use Listeners annotation (@Listeners) as shown below:
- @Listeners(Week1.CustomReport.class)



```
package Week1;

import org.testng.annotations.Test;

@Listeners(Week1.CustomReport.class)
public class SDET_Pro {

 //WebDriver driver ;
```

Second way is to use the same in XML file before test class, Refer below example:



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd"
<suite name="Suite">

   <listeners>
   <listener class-name="Listener_Demo.ListenerTest"/>
   </listeners>

  <test name="Test">
     <classes>
       <class name="Listener_Demo.TestCases"/>
     </classes>
   </test> <!-- Test -->
</suite> <!-- Suite -->
```

LAB 6

Scenario 1- Login to OrangeHRMlive- https://opensource-demo.orangehrmlive.com/ in a Test method

- Validate page title to be “OrangeHRM”
- Verify that dashboard menu is available
- print page title in iTestListner

Scenario 2- Login to OrangeHRMLive application-

- Click on Admin
- Search UserName = Newboo and validate searchresult is returning record.

Note- Use property file

TestNG -
Retry Analyzer

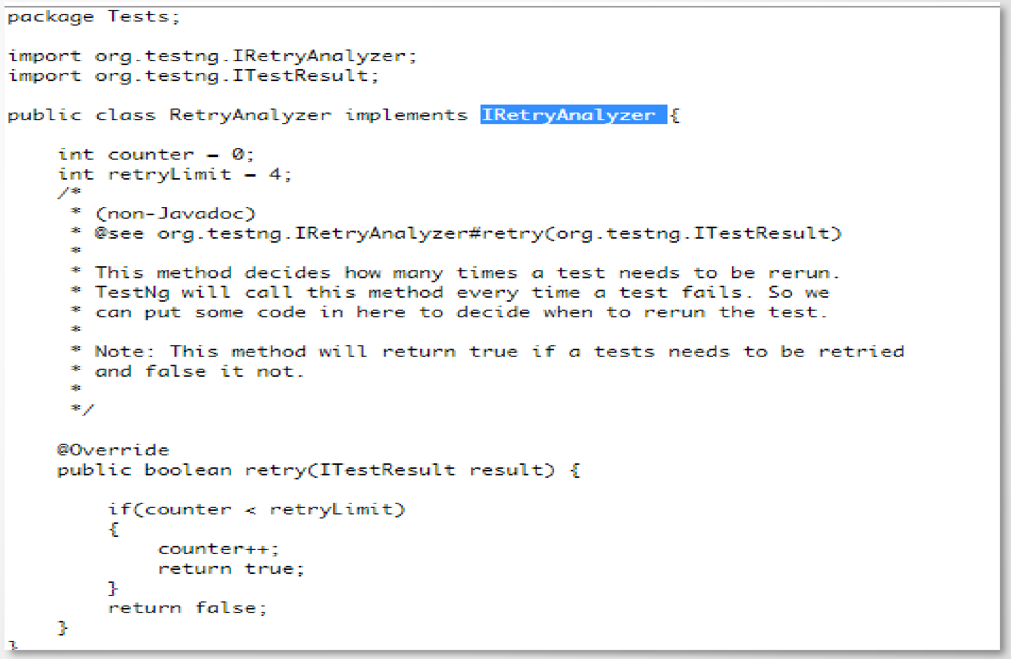
While writing selenium test using testNG user must have seen random failure during an automated test run. These failures might not necessarily be because of product bugs. These failure can be because of the following reasons -

- Random browser issues or browser becoming unresponsive
- Random machine issues
- Server issues like unexpected delay in the response from server.

Implementation - Retry Analyzer 1/2

Step 1- Create a class give any name, implement IRetryAnalyzer.

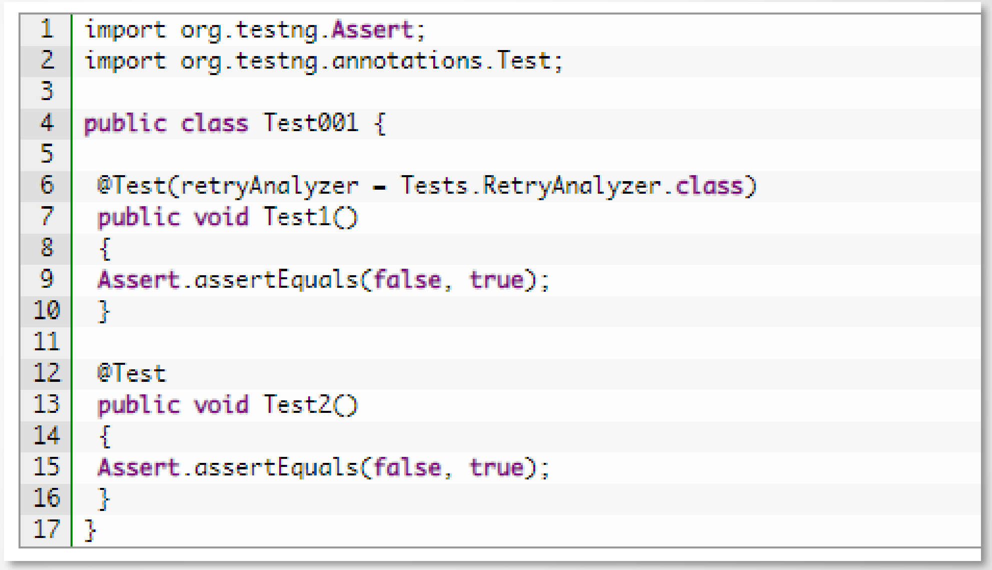
Refer below screenshot, mention values for retryLimit count as per need



```
package Tests;

import org.testng.IRetryAnalyzer;
import org.testng.ITestResult;

public class RetryAnalyzer implements IRetryAnalyzer {

    int counter = 0;
    int retryLimit = 4;
    /*
     * (non-Javadoc)
     * @see org.testng.IRetryAnalyzer#retry(org.testng.ITestResult)
     *
     * This method decides how many times a test needs to be rerun.
     * TestNg will call this method every time a test fails. So we
     * can put some code in here to decide when to rerun the test.
     *
     * Note: This method will return true if a tests needs to be retried
     * and false it not.
     *
     */

    @Override
    public boolean retry(ITestResult result) {

        if(counter < retryLimit)
        {
            counter++;
            return true;
        }
        return false;
    }
}
```

Implementation - Retry Analyzer 2/2

Step 2- Implement We can do this by simply using following syntax to @Test(retryAnalyzer="IRetryAnalyzer Implementing class"). Below is the code to do that



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

public class Test001 {

 @Test(retryAnalyzer = Tests.RetryAnalyzer.class)
 public void Test1()
 {
 Assert.assertEquals(false, true);
 }

 @Test
 public void Test2()
 {
 Assert.assertEquals(false, true);
 }
}
```

LAB 7

1. Write custom/override method for onTestFailure and onTestSuccess.

   A- Print Page title onTestSuccess
   B- In @Test method mention Assert.fail(), and print launched url(using value passing across class.

2. Fail a test method and try to rerun it using retry analyzer counter

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

ATDD

ATDD - Definition

In this article, we highlight the similarities and differences between two popular testing methods commonly known as TDD and ATDD.

TDD stands for Test-Driven Development, while ATDD stands for Acceptance Test-Driven Development. Understanding how these two testing approaches work is critical for testing professionals and this post will be a primer to get you started on your discovery of both.

What is Test-Driven Development (TDD)?

TDD is a system of developing software following Extreme Programming (XP) principles, however over time it spun off as an independent software development technique.

What is Acceptance Test-Driven Development (ATDD)?
ATDD is a collaborative method of testing which forces all the people involved in the creation of new software (e.g. testers, developers and users) to define as a team the acceptance criteria that the system has to fulfil in the early stages of development.

Cucumber

Definition -

Cucumber is a tool based on Behavior Driven Development (BDD) framework which is used to write acceptance tests for the web application.

It allows automation of functional validation in easily readable and understandable format (like plain English) to

- ❑ Business Analysts
- ❑ Developers
- ❑ Testers, etc

Cucumber - Configuration

Steps to follow:

- Launch the Eclipse IDE and from Help menu, click “Install New Software”.
- You will see a dialog window, click “Add” button.
- Type name as you wish, let’s take “Cucumber” and type “http://cucumber.github.com/cucumber-eclipse/update-site” as location. Click OK.
- You come back to the previous window but this time you must see Cucumber Eclipse Plugin option in the available software list. Just Check the box and press “Next” button.
- Click on Next, Click “I accept the terms of the license agreement” then click Finish.
- Let it install, it will take few seconds to complete.
- You may or may not encounter a Security warning, if in case you do just click OK.
- Done

Cucumber - Configuration

Feature file: A feature would describe the current test script which has to be executed.

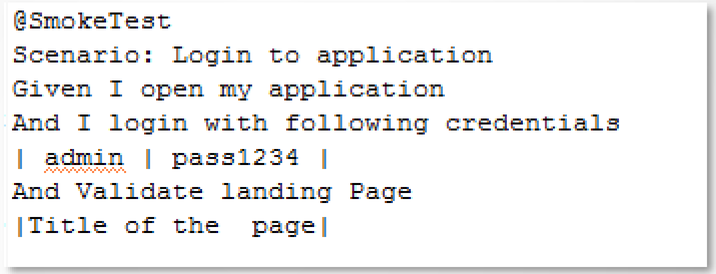
- Scenario: Scenario describes the steps and expected outcome for a particular test case.
- Scenario Outline: Same scenario can be executed for multiple sets of data using scenario outline. The data is provided by a tabular structure separated by (I I).
- Given: It specifies the context of the text to be executed. By using datatables "Given", step can also be parameterized.
- When: "When" specifies the test action that has to performed
- Then: The expected outcome of the test can be represented by "Then"

Test Runner file: Test runner file is a bridge between feature file and step definition. Testrunner file drives cucumber flow and publish reports.

Step Definition: Step definition file contains detail code flow of selenium interaction with web.

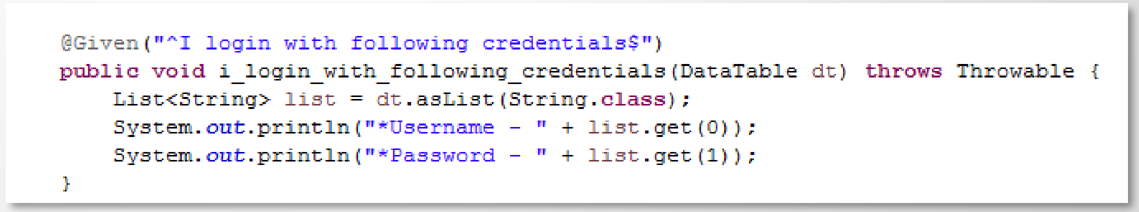
Cucumber

Feature file:



```
@SmokeTest
Scenario: Login to application
Given I open my application
And I login with following credentials
| admin | pass1234 |
And Validate landing Page
|Title of the  page|
```

Step definition:



```
@Given("^I login with following credentials$")
public void i_login_with_following_credentials(DataTable dt) throws Throwable {
    List<String> list = dt.asList(String.class);
    System.out.println("*Username - " + list.get(0));
    System.out.println("*Password - " + list.get(1));
}
```

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

Cucumber - Scenario Outline

Feature file:

Scenario outline basically replaces variable/keywords with the value from the table. Each row in the table is considered to be a scenario.

```
Scenario Outline: Create ABC
Given I open the application
When I enter username as “<username>”
And I enter password as “<password>”
Then I enter title as “<title>”
And press submit
Examples:  | username | password | title |
           | Rob | xyz1 | title1 |
           | Bob | xyz1 | title2 |
```

Cucumber - Data table and Scenario Outline

Difference between Scenario Outline & Data Table

Scenario Outline:
This uses Example keyword to define the test data for the Scenario
This works for the whole test
Cucumber automatically run the complete test. i.e., The number of times equal to the number of data in the Tet Set

Test Data*:*
No keyword is used to define the test data
This works only for the single step, below which it is defined
A separate code is required to understand the test data which can run for single or multiple times.

Cucumber - Hooks

What are Hooks in Cucumber?

- Cucumber supports hooks, which are blocks of code that run before or after each scenario. You can define them anywhere in your project or step definition layers, using the methods @Before and @After.

- Cucumber Hooks allows us to better manage the code workflow and helps us to reduce the code redundancy. We can say that it is an unseen step, which allows us to perform our scenarios or tests.

Cucumber - Sample Hooks

```
package utilities;
import cucumber.api.java.After;
import cucumber.api.java.Before;

public class Hooks {
@Before
   public void beforeScenario(){
      System.out.println("This will run before the Scenario");
   }
@After
   public void afterScenario(){
      System.out.println("This will run after the Scenario");
   }
}
```

Cucumber - Background

- Background in Cucumber is used to define a step or series of steps that are common to all the tests in the feature file.
- It allows you to add some context to the scenarios for a feature where it is defined.
- A Background is much like a scenario containing a number of steps But it runs before each and every scenario for a feature in which it is defined.

```
Feature: Test Background Feature
Description: The purpose of this feature is to test the Background keyword

Background: User is Logged In
Given I navigate to the login page
When I submit username and password
Then I should be logged in

Scenario: Search a product in Ecommerce site
Given User search for Keyboard
When Add the keyboard to the Cart
Then Cart should display with Keyboard
```

LAB 8

1. Write Cucumber feature and step definition for below condition.
   Scenario: Login to orange HR application
   Given I open my application
   And I login with following credentials
   | admin | admin123 |
   And Validate landing Page
   |Title of the page|

2. Write Step definition for below GWT Condition.
   Scenario: The sum of a list of numbers should be calculated
   Given a list of numbers
   | 4 |
   | 5 |
   | 6 |
   When I Multiply them
   Then should I get 120

Selenium

Maven

Maven - Definition

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. Using maven we can build and manage any Java based project. This tutorial will teach you how to use Maven in your day-to-day life of any project development using Java

Maven provides developers ways to manage the following –

- Builds
- Documentation
- Reporting
- Dependencies
- SCMs
- Releases
- Distribution
- Mailing list

Maven - Configuration

Download Maven 2.2.1 from https://maven.apache.org/download.cgi

1. Set the environment variables using system properties.

M2_HOME=C:\Program Files\Apache Software Foundation\apache-maven-3.3.1
M2=%M2_HOME%\bin MAVEN_OPTS=-Xms256m -Xmx512m

Steps to Install:

- In Eclipse IDE, select Help | Install New Software from Eclipse Main Menu.
- On the Install dialog, Enter the URL http://download.eclipse.org/technology/m2e/releases/. Select Work with and m2e plugin as shown in the following screenshot:
- Click on Next button and finish installation.
- Configure Eclipse with Maven
- With m2e plugin is installed, we now need create Maven project.
- In Eclipse IDE, create a new project by selecting File | New | Other from Eclipse menu.
- On the New dialog, select Maven | Maven Project and click Next
- On the New Maven Project dialog select the Create a simple project and click Next
- Enter WebdriverTest in Group Id: and Artifact Id: and click finish.

Maven- Sample POM

```
<project xmlns = "http://maven.apache.org/POM/4.0.0" xmlns:xsi =
"http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation =
"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd"> <modelVersion>4.0.0</modelVersion>
<groupId>com.companyname.project-group</groupId>
<artifactId>project</artifactId>
<version>1.0</version>
<dependencies>
<dependency>
</dependency>
</dependencies>
</project>
```

How to find Maven dependency of any jar file- Visit mvnrepository.com there all dependency (with version) is listed

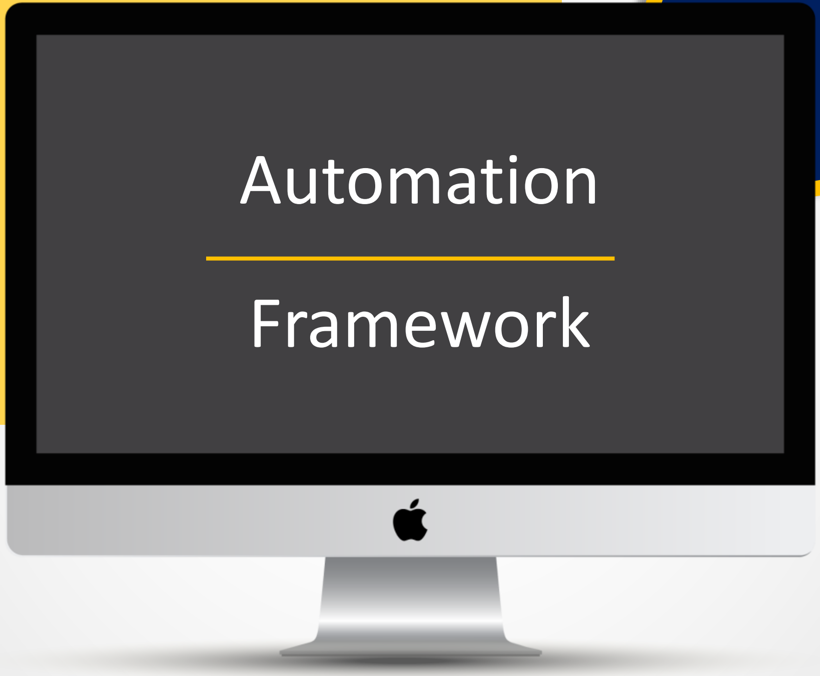
Sample dependency for TestNG- https://mvnrepository.com/artifact/org.testng/testng

LAB 9

1. Write TestNg Dependency in Maven – pom.xml file

2. Write a test for launching google.com in IE Browser, Using Test NG annotation @BeforeTest, @Test, @AfterTest annotation

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED



Automation Framework

Automation framework – Definition

Automation framework- Automation framework is how we frame our work, which includes :

- Test data - Where we keep it and how we are using it
- Scripting logic - Quality of scripts in terms of reusability
- Reporting - Detailed and Consolidated execution logs can help users to understand the execution status

Automation Framework – Types

There are majorly 6 types of framework exist –

- ✓ Linear Scripting
- ✓ The Test Library Architecture Framework.
- ✓ The Data-Driven Testing Framework.
- ✓ The Keyword-Driven or Table-Driven Testing Framework.
- ✓ The Hybrid Test Automation Framework.
- ✓ Behaviour Driven Development

Automation Framework – Planning

In most of the organisation, they have 2 separate teams or logical segregation between them-

1- Script creation for new Functionality
2- Script Maintenance

Estimation in any automation project is done largely done on following key points-

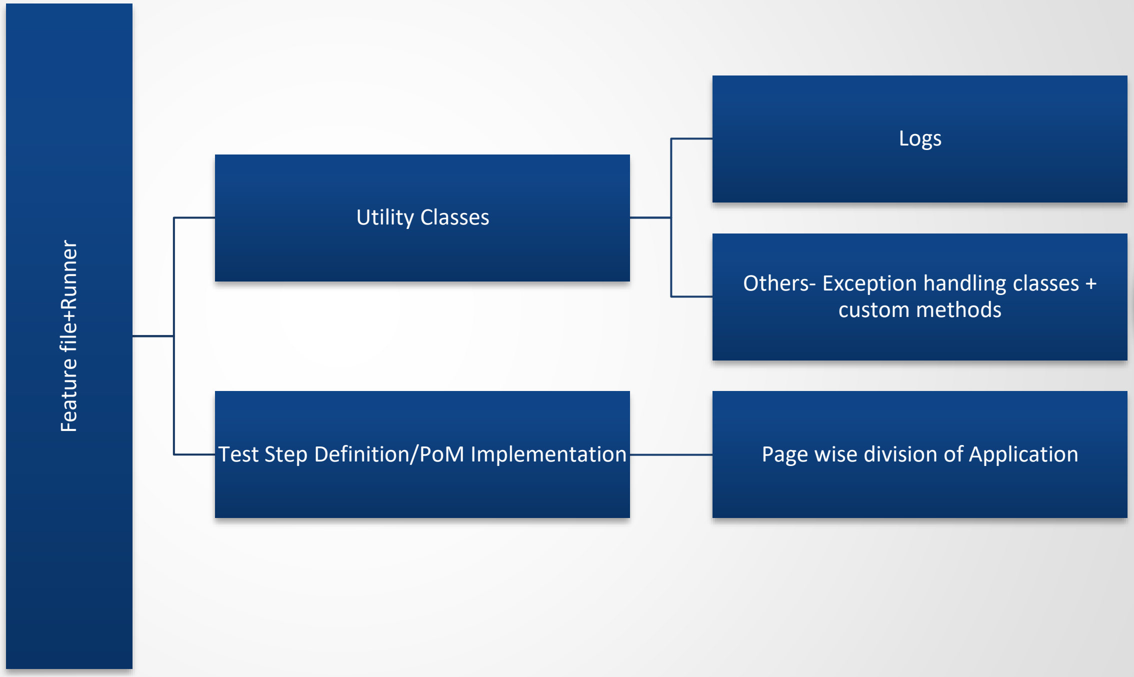
- Application technology and functionality(complexity)
- Framework usability and features

Automation Framework – Implementation

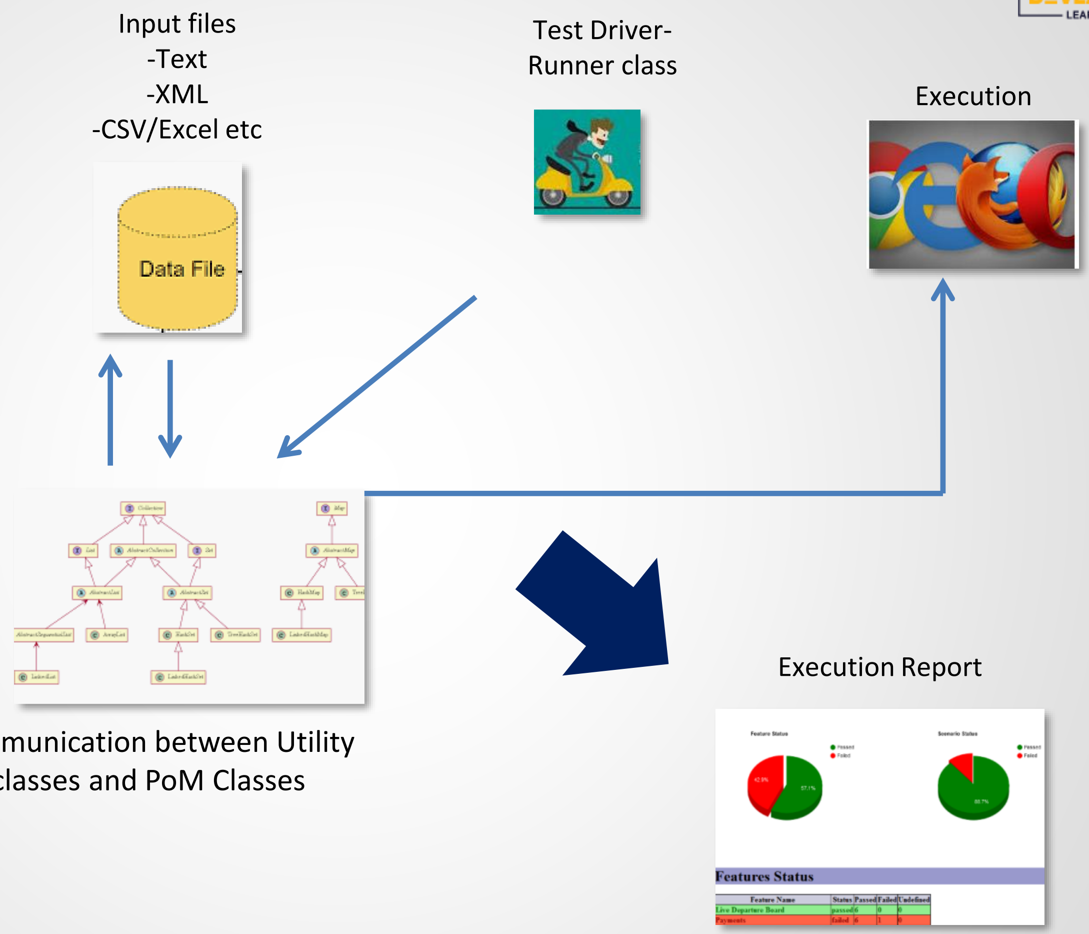
We will Implement selenium framework that comprises of -

- Cucumber
- Junit
- Code Layer
  - ✓ Utility classes
  - ✓ Reusable methods- Application functionality implemented in Page Object Model
- Reporting- Extent Reports

Automation Framework – Implementation..1/2


Feature file+Runner
Utility Classes
Logs
Others- Exception handling classes + custom methods
Test Step Definition/PoM Implementation
Page wise division of Application

Automation Framework – Implementation..2/2



Features Status

Feature Name	Status	Passed	Failed	Undefined
Live Departure Board	passed	6	0	0
Payments	failed	6	1	0

Page Object Model

- Page Object Model is a design pattern to create Object Repository for web UI elements.
- Under this model, for each web page in the application, there should be corresponding page class. This Page class will find the WebElements of that web page and also contains Page methods which perform operations on those WebElements.
- Name of these methods should be given as per the task they are performing, i.e., for a login page, we need to create login class where web object properties needs to be mentioned and then methods to be written in same class.

Page Object Model - Benefits

In Page Object Pattern test flow, related methods and verification needs to be implemented separately, which results in to easier maintenance.

Its Benefits are -

- Is independent of test cases, so we can use the same object repository for a different purpose with different tools. For example, we can integrate POM with TestNG/JUnit for functional Testing and at the same time with JBehave/Cucumber for acceptance testing.
- Code becomes less and optimized because of the reusable page methods in the POM classes.
- Methods get more realistic names which can be easily mapped with the operation happening in UI. i.e. if after clicking on the button we land on the home page, the method name will be like 'gotoHomePage()'.

LAB 10

1. Write Page object class for Directory menu
2. Write Page object class for Admin menu

Q&A

DEVLABS ALLIANCE®
LEARN, LEAD & SUCCEED

THANK YOU