Testing Masters

TECHNOLOGIES

1. Selenium Overview

Introduction

Selenium is an open-source and a portable automated software testing tool for testing web applications. It has capabilities to operate across different browsers and operating systems. Selenium is not just a single tool but a set of tools that helps testers to automate web-based applications more efficiently.

Advantages of Selenium

UFT/QTP and Selenium are the most used tools in the market for software automation testing. Hence it makes sense to compare the pros of Selenium over UFT/ QTP.

Feature	Selenium	QTP/UFT
License Cost	Open source tool and Zero Cost	Commercial tool and huge license cost
Browser Support	Firefox, Chrome, IE, Safari, Opera, etc.,	Firefox, Chrome, IE
OS Support	Windows, MAC, Unix	Windows
Language Support	Java, C#, Ruby, Python, etc.,	VB Scripting
Parallel Execution	Supports	Does not Support
Remote Execution	Supports[Using Grid]	Does not support
Hardware utilization	Low	High

2. Selenium Components:

The following are the components of selenium

* Selenium IDE
* Selenium RC
* Selenium WebDriver
* Selenium Grid

Selenium IDE:

- ✓ Selenium IDE Stands for Integrated Development Environment.
- ✓ Selenium IDE is a Firefox plug in that lets testers to record their actions as they follow the Workflow that they need to test.
- ✓ Selenium does not support any programming language.
- ✓ To overcome this Selenium RC is introduced.

Selenium RC:

- ✓ Selenium RC stands for Selenium Remote Control(Selenium 1.0 version)
- ✓ Selenium RC supports programming languages like Java, C# etc.,

- ✓ But in Selenium RC the written Test Script code first interacts with Server and then interacts with webpage.
- ✓ To overcome these architectural drawbacks selenium Webdriver is introduced.

Selenium WedDriver:

- ✓ Selenium WebDriver [Selenium 2.0 Version] is the successor to Selenium RC [Selenium 1.0 Version].
- ✓ Selenium WebDriver will sends commands directly to the browser and retrieves results.
- ✓ Selenium WebDriver has Robust and powerful methods to can be used easily.

Selenium Grid:

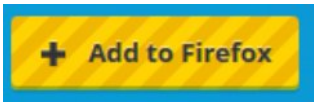
- ✓ Selenium Grid is a toll that can be used for Remote and parallel Execution.
- ✓ Selenium Grid can be configured with both RC, Webdriver versions.
- ✓ Using Selenium Grid will help in reducing the execution time drastically.

3. Selenium IDE

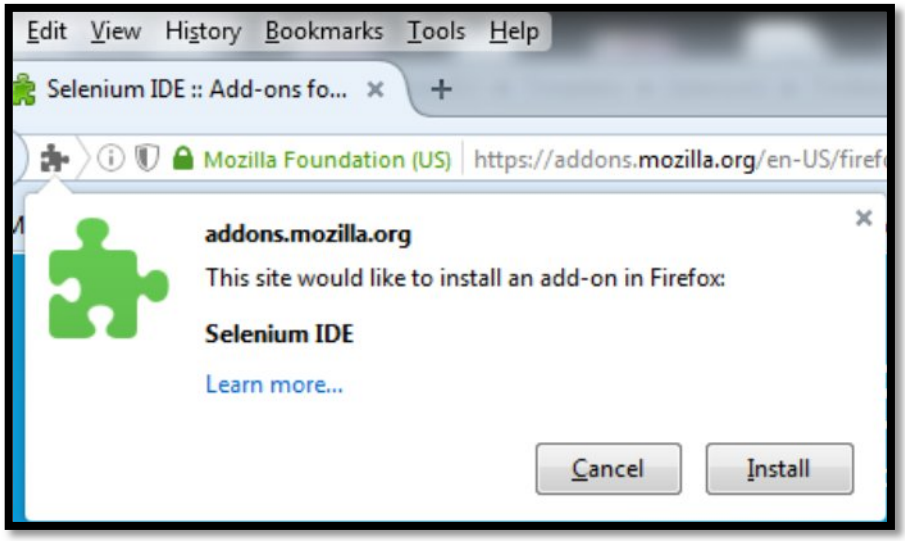
- ✓ Selenium IDE stands for selenium Integrated Development environment.
- ✓ Selenium IDE is launched in year 2006 by thought works organization, in competition to the QTP tool of mercury.
- ✓ Selenium IDE does not have any programming language support.
- ✓ Selenium IDE is an add-on for Firefox browser.
- ✓ In Selenium IDE, we need to enter the commands and object properties to perform actions.
- ✓ Selenium IDE does not support good reporting formats.
- ✓ Because of this drawback, the other components of selenium are introduced.
- ✓ However, the recorded scripts can be converted into various programming languages supported by Selenium and the scripts can be executed on other browsers as well.

3.1 Installation of selenium IDE:

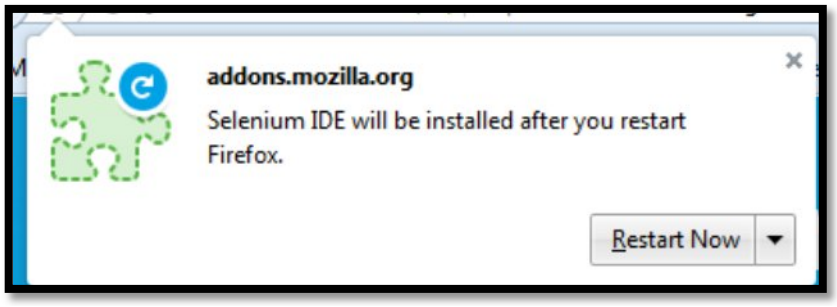
- ✓ Open a Firefox browser
- ✓ Navigate to this URL *https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/* [Google: Selenium IDE for Firefox]
- ✓ Click on add to Firefox button



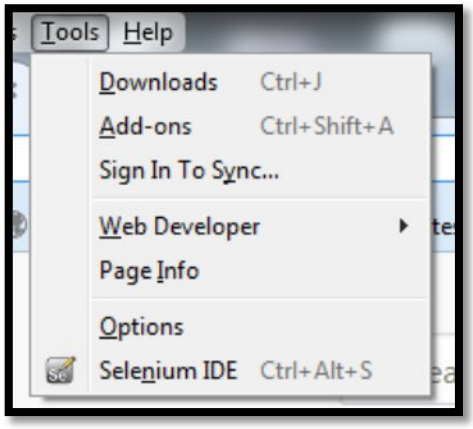
- ✓ After downloading, click on install button at the top, to install selenium IDE.



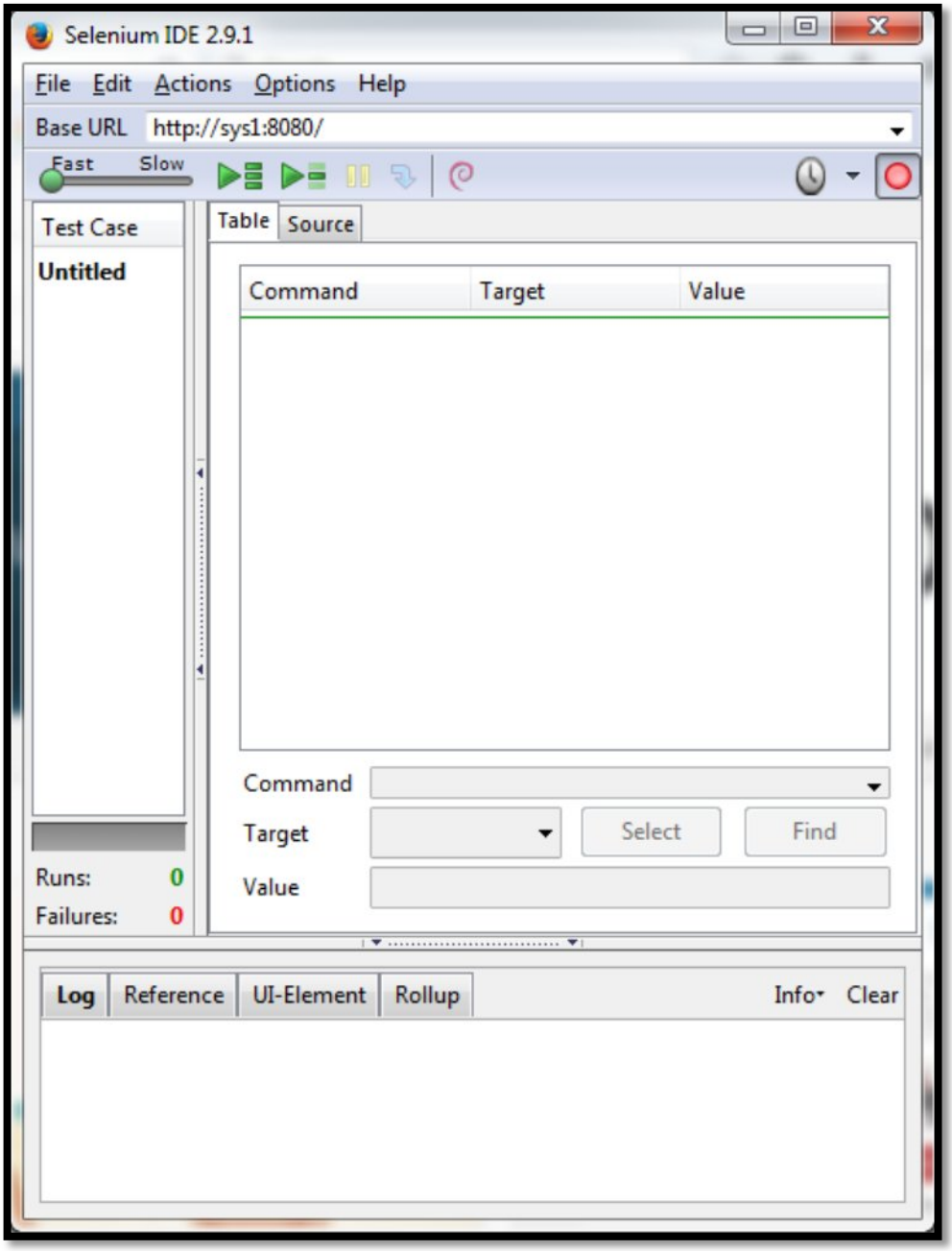
✓ After Installation, Restart the Firefox browser.



✓ After successful installation, selenium IDE will be visible in Tools Selenium IDE.



✓ The Icon will also be visible at the top right corner.
✓ Click on this Icon to open Selenium IDE.



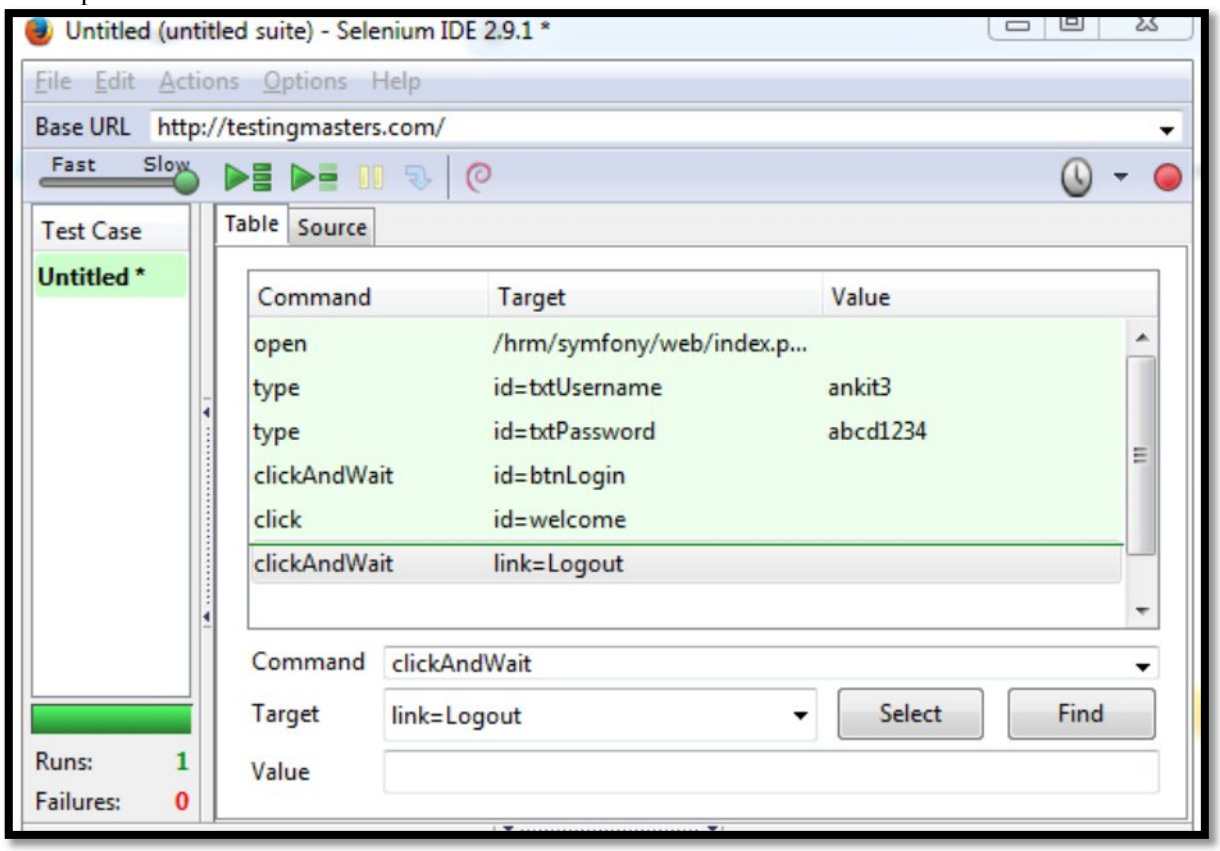
3.2 Working with Selenium IDE:

- ✓ Open Selenium IDE[Tools Selenium IDE]
- ✓ Open the AUT{Application Under Test}
- ✓ Start performing some actions in the webpage.
- ✓ All the Actions along with data are recorded in the form of IDE Table.

- ✓ Selenium IDE Table

Command	Target	Value
Selects the command based on action performed	Builds the locator Description based on the Web element Selected	Stores the data based on the values entered.

- ✓ Example:



- ✓ File Save Test Case as Give the test case name and save the test case.
- ✓ File Export Test Case As Select the Appropriate version click on Save.

4. Selenium WebDriver:

4.1 Environment Setup for Selenium WebDriver:

In order to develop WebDriver scripts, users have to ensure that they have the initial Configuration done. Setting up the environment involves the following steps.

- ❖ Download and Install Java
- ❖ Download and Configure Eclipse
- ❖ Download and Configure Firebug and Firepath
- ❖ Configure Selenium WebDriver
- ❖ First WebDriver Program

4.1.1: Download and Install Java

- ✓ Navigate to "http://www.oracle.com/technetwork/java/javase/downloads/index.html"
- ✓ [Google: Java JDK Download]
- ✓ Click on JDK Download.
- ✓ Click on accept license agreement radio button.
- ✓ Download the JDK file based on the system type
- ✓ [Right click on My computer Properties System type]
- ✓ 64 bit -- Windows X64 -- click on "jdk-8u101-windows-x64.exe"
- ✓ 32 bit -- Windows X86 -- Click on "jdk-8u101-windows-i586.exe"
- ✓ The " jdk-8u77-windows-x64.exe" JDK will be downloaded on clicking above link
- ✓ Double click on this exe file and proceed install Java.
- ✓ After successful installation we can view JDK,JRE files in the below location
  "C:\Program Files\Java"

4.1.2: Download and Configure Eclipse

- ✓ Download Eclipse from the link.
  - ❖ *https://drive.google.com/file/d/0B5fakF7Z_ekCN2lJTml5U1JNdjg/view?usp=sharing*
- ✓ Zip file will be downloaded.
- ✓ Unzip this file to extract actual file contents.
- ✓ Go to the extracted file[eclipse-jee-neon-1-win32-x86_64Eclipseclick on eclipse.exe file]
- ✓ On clicking on this exe file, a window will be opened to specify the workspace location
- ✓ Workspace location specifies the hard disk location where to store codes.
- ✓ Click on OK after specifying the work space location.
- ✓ Click on Workbench icon [Displayed at the Right top corner]

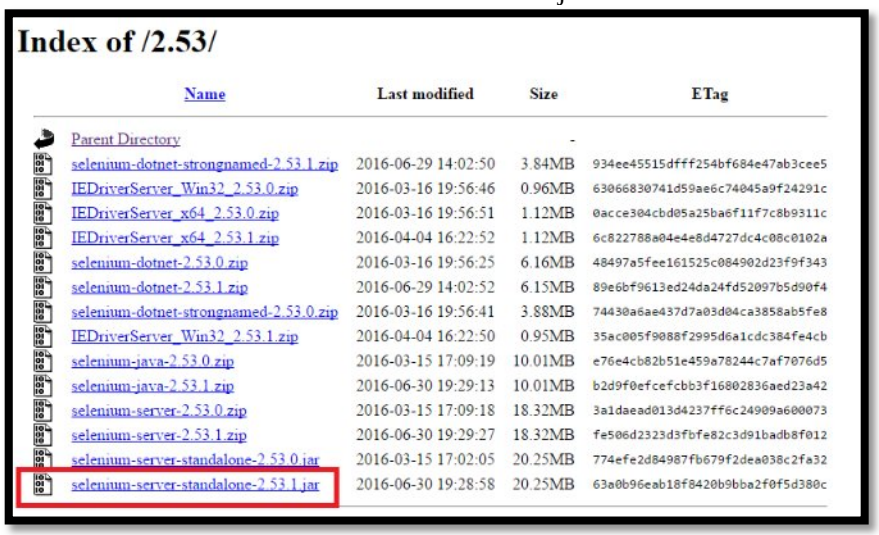
4.1.3: Download and Configure Firebug and Firepath

- ✓ Firebug add-on is supported only for Firefox browsers.
- ✓ The following are the steps to install firebug add-on.
  - ❖ Open Firefox browser
  - ❖ Navigate to "https://addons.mozilla.org/en-US/firefox/addon/firebug/" URL. [Google: firebug for Firefox]
  - ❖ click on 'Add to Firefox' Button
  - ❖ Click on install button after downloading.
  - ❖ after successful installation, firebug can be viewed in following path
    - ▪ Tools Web Developer Firebug
    - ▪ An icon will be displayed at the top right corner.
  - ❖ This completes the firebug installation.
- ✓ Firepath is an sub add-on for firebug.
- ✓ The following are the steps to install Firepath add-on.
  - ❖ Open Firefox browser
  - ❖ Navigate to "https://addons.mozilla.org/En-us/firefox/addon/firepath/" URL. [Google: Firepath for Firefox]
  - ❖ click on 'Add to Firefox' Button
  - ❖ Click on install button after downloading.

- ❖ After successful installation, Restart the Firefox browser.
- ❖ This completes the firebug installation.

4.1.4: Configure Selenium WebDriver:

- ✓ Download the selenium server standalone.jar file from the below URL
  http://selenium-release.storage.googleapis.com/index.html?path=2.53/
- ✓ Click on Selenium-server-standalone-2.53.1.jar



- ✓ Download the supported browser version accordingly.
- ✓ Associate this jar file with current Project in eclipse
  - ➢ Open eclipse
  - ➢ Create a project
  - ➢ Associate selenium jar file to the current project.
    Right Click on java Project Build Path Configure build path Libraries Add External Jars Browse for jar file.
  - ➢ Create the object for the respective browser driver class and use the required methods.

4.1.5 First Webdriver Program:

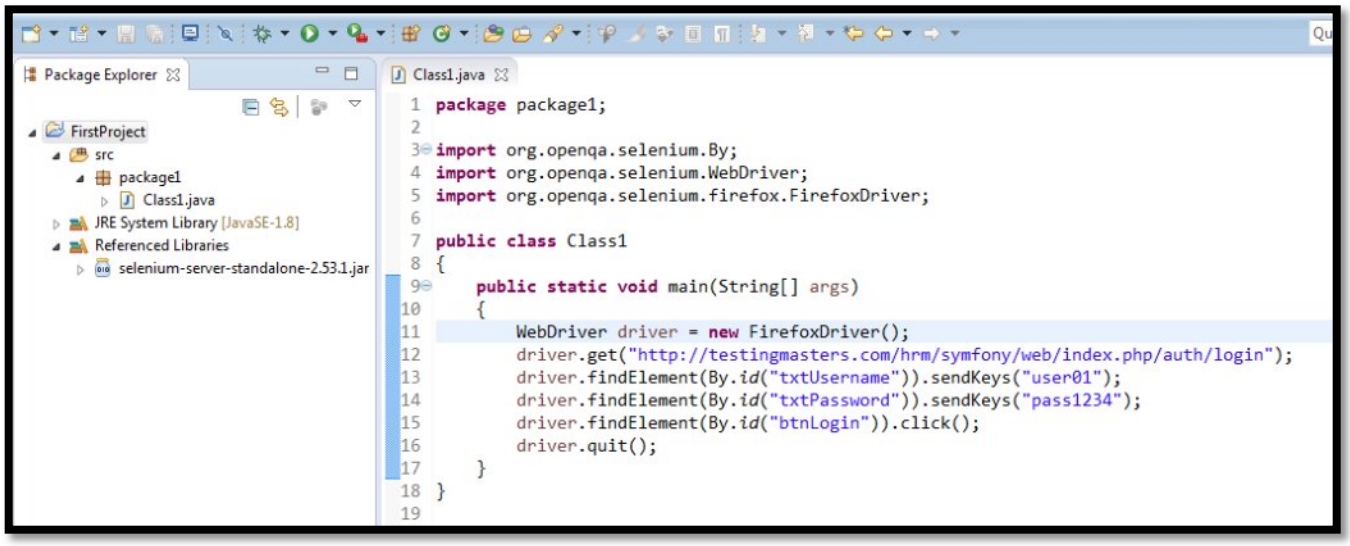
Pre Checks:

- ❖ Java is installed and should be on or above 1.8 version.
- ❖ Eclipse should be on or above Mars version.
- ❖ Selenium Webdriver version should be on or above 2.53.1 version.

Steps for First Selenium WebDriver Program:

- ✓ Open eclipse
- ✓ Create a project -> Package -> Class
- ✓ Associate selenium jar file to the current project.
- ✓ Identify the Xpath or Locator of Each Web element
- ✓ Use this xpath/locator for developing code in the webdriver.

Example:



4.2 WebDriver Methods:

Selenium webdriver methods are used to perform operations on the browser and web elements. Using Web element locators and Webdriver methods the test scripts can be written.

The following are the webdriver methods:

- ❖ get()
- ❖ navigate()
- ❖ findElement()
- ❖ findElements()
- ❖ getCurrentURL()
- ❖ getTitle()
- ❖ getWindowHandle()
- ❖ getWindowHandles()
- ❖ close()
- ❖ quit()

get():

- ✓ This method is used to launch a URL in the current driver instance.
- ✓ This method will launch the URL and waits until all the elements of a webpage are loaded.

Syntax:

driver.get(Fully Qualified URL)

Example:

driver.get("https://www.google.co.in")

Note:

Fully qualified URL will start with https or http.

navigate():

- ✓ This method is also used to launch a url in the driver instance and waits for all the elements to be loaded.

- ✓ But this method also supports, the features such as
  - ❖ Backward
  - ❖ forward
  - ❖ refresh

Example:

```
WebDriver driver = new FirefoxDriver();
driver.navigate().to("https://talentzing.com/");
driver.findElement(By.xpath("//span[@id='lnkRegistration']")).click();
driver.findElement(By.xpath("//span[@id='lblJsRegister']")).click();
driver.navigate().refresh();
driver.navigate().back();
```

Note:

- ➢ back(), forward() methods will navigate to the pages based on the history of the page navigations in the driver instance.

FindElement():

- ✓ This method is used to find a web element in the current document of driver instance based on the node properties.
- ✓ the following are the sub methods that findElment supports
  - ❖ sendKeys – to enter text
  - ❖ click – to perform click operation
  - ❖ clear – to clear the existing contents
  - ❖ isDisplayed – to verify the existence – true/false
  - ❖ isEnabled – to verify if the field is enabled – true/false
  - ❖ isSelected – if the checkbox is selected – true/false
  - ❖ getAttribute – to get the attribute value/property value.

Example1:

```
driver.findElement(By.xpath("//input[@name='userName']")).sendKeys("abcd123");
```

Example2:

```
driver.findElement(By.xpath("//input[@name='Login']")).click();
```

Example3:

```
driver.findElement(By.xpath("//input[@name='userName']")).clear();
driver.findElement(By.xpath("//input[@name='userName']")).sendKeys("abcd123");
```

Example4:

```
boolean flagdisplayed;
flagdisplayed = driver.findElement(By.xpath("//input[@name='userName']")).isDisplayed();
System.out.println(flagdisplayed);
```

Example5:

```
boolean flagEnabled;
flagEnabled = driver.findElement(By.xpath("//input[@name='userName']")).isEnabled();
System.out.println(flagEnabled);
```

Example6:

```
boolean flagSelected;
flagSelected = driver.findElement(By.xpath("//input[@name='checkbox']")).isSelected();
System.out.println(flagSelected);
```

Example7:

```
String value = "";
value = driver.findElement(By.xpath("//input[@name='login']")).getAttribute("width");
System.out.println(value);
```

Note:

- ✓ If the element description is matching more than one node, by default this method will points to the first occurrence.
- ✓ In order to point out to specific occurrence, we need to use index in xpath.

findElements():

- ✓ This method is used to perform action on a group of web elements.
- ✓ This method will return the web elements in the form of a list.
- ✓ From the list, we have to iterate using for each loop and then perform action on each webelement.

  Syntax:

  List<WebElement> objectname = driver.findElements(by.Xpath("xpath Expression"));
- ✓ To get the number of matching web elements, we can use objectname.size();

Program to get the names of all the radio buttons present on a webpage:

```
WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.get("http://newtours.demoaut.com/");
driver.findElement(By.name("userName")).sendKeys("mercury");
driver.findElement(By.name("password")).sendKeys("mercury");
driver.findElement(By.name("login")).click();

List<WebElement> allitems = driver.findElements(By.xpath("//input[@type='radio']"));
for(WebElement ele:allitems)
{
        System.out.println(ele.getAttribute("name"));
}
```

Program to get the number of links displayed in a webpage:

```
WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
driver.get("http://newtours.demoaut.com/");
List<WebElement> allitems = driver.findElements(By.xpath("//a"));
System.out.println(allitems.size());
```

Program to get the names of all the radiobuttons:

```
WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
driver.get("http://newtours.demoaut.com/");
driver.findElement(By.name("userName")).sendKeys("mercury");
driver.findElement(By.name("password")).sendKeys("mercury");
driver.findElement(By.name("login")).click();
List<WebElement> allitems = driver.findElements(By.xpath("//input[@type='radio']"));
for(WebElement ele:allitems)
{
        System.out.println(ele.getAttribute("name"));
}
```

getCurrentURL():
This method is used to get the current URL of the webdriver instance.
Example:

```
System.out.println(driver.getCurrentUrl());
```

getTitle():
This method is used to get the title of the webdriver instance.
Example:

```
System.out.println(driver.getTitle());
```

getwindowhandle():
This method is used to get the runtime generated page id of the driver instance.
Example1:

```
System.out.println(driver.getWindowHandle());
```

Example2:

```
WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.manage().window().maximize();
driver.get("http://newtours.demoaut.com/");
System.out.println(driver.getCurrentUrl());
System.out.println(driver.getTitle());
System.out.println(driver.getWindowHandle());
```

Output:

```
http://newtours.demoaut.com/
Welcome: Mercury Tours
{38ed9a6c-dedc-4d73-a86d-1242d22f0470}
```

getwindowhandles():

- ✓ This method is used to get all the window handles of the diver instance.
- ✓ The window handles will be returned in the form a String set.

Code to get the window handles:

```
Set<String> allhandles = driver.getWindowHandles();
for(String str1:allhandles)
{
        System.out.println(str1);
}
```

close():

- ✓ Close method is used to close only the current window of the driver instance.

quit():

- ✓ quit method will close all the windows of the driver instance.

4.3 Switch To Window:

- ✓ Every web page will be having URL, Title, Window Handle, Document.
- ✓ Driver object can point out to only one webpage/document at a time.
- ✓ By default driver object will be pointing out to parent window.
- ✓ In order to make the driver object to point out to child window we need to use the handle of the window.

Syntax:
driver.switchTo().window(WindowHandle);

- ✓ The following are the steps to be followed to switch the control to child window and get back to parent window
  - ❖ get the parent window handle & Store in a variable
  - ❖ get all the window handles
  - ❖ switch to each window by using the handles
  - ❖ get Title/URL of the window and verify
  - ❖ Perform desired actions on child window.
  - ❖ close the child window
  - ❖ Switch back the control to parent window.

Example:

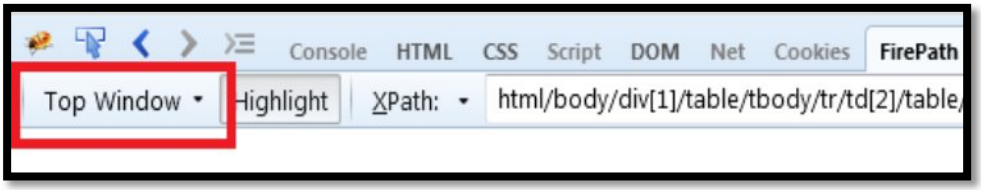
```
WebDriver driver = new FirefoxDriver();
driver.get("https://talentzing.com/");
driver.findElement(By.xpath("//a[text()='Terms & Conditions']")).click();
driver.findElement(By.xpath("//a[text()='FeedBack']")).click();
String parentHandle;
parentHandle = driver.getWindowHandle();
Set<String> allhandles = driver.getWindowHandles();
for(String h1:allhandles)
{
        driver.switchTo().window(h1);
        String URL = driver.getCurrentUrl();
        if (URL.contains("TermsAndConditions"))
        {
                driver.findElement(By.xpath("//input[@id='btnOk']")).click();
                driver.close();
                break;
        }
}
driver.switchTo().window(parentHandle);
```

4.4 Switch to Frame:

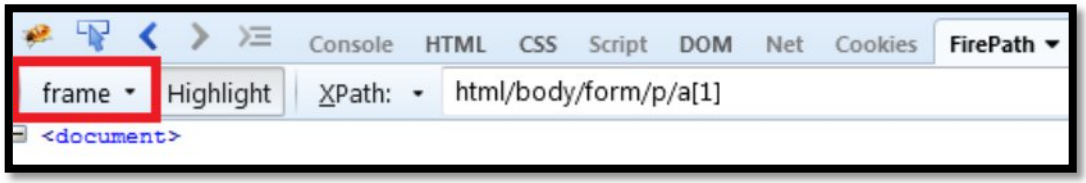
- ✓ A frame is document that is defined inside a parent document.
- ✓ As the driver object can point to only one frame at a time, by default it points to the parent document.
- ✓ So, inorder to perform any operation on the elements in the frame, first we need to switch to the respective frame.
- ✓ Firepath view can handle only one document at a time.
- ✓ In Order view the complete set of documents associated with current webpage, we need to move to HTML View.

The following is the process to identify a frame.

- ✓ While spying an object in normal page using firepath,in the top left corner, it will show as Top Window.



- ✓ But, while spying an object which is inside a frame, it would look like,



- ✓ Now, Select the HTML View, instead of firepath view and select the object.
- ✓ Traverse the parent hierarchy of the selected node to identify the frame details.

- ✓ Using this frame details, but the web element object.

WebElement ele = driver.findElement(By.Xpath("//frame[@name='login_Page]"))

The following are the steps to switch to the frame

- ✓ Create a webelement object to the respective frame.
- ✓ Use this webelement object to switch to the frame.
- ✓ Perform the desired operation in the frame.
- ✓ Switch back to the parent document.

Code to switch to frame:

```
WebElement frameobj = driver.findElement(By.xpath("//frame[@name='login_page']"));
driver.switchTo().frame(frameobj);
```

Code to switchback to parent document:

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

Note:
If two frames are present adjacently, then

1. first we need to switch to one frame
2. switch to default content
3. switch to the second frame.

Example1:

```
WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.navigate().to("https://netbanking.hdfcbank.com/netbanking/");
WebElement ele = driver.findElement(By.xpath("//frame[@name='login_page']"));
driver.switchTo().frame(ele);
driver.findElement(By.xpath("//img[@alt='continue']")).click();
driver.switchTo().defaultContent();
```

Example 2:

```
WebDriver driver = new FirefoxDriver();
driver.get("https://netbanking.hdfcbank.com/netbanking/");
WebElement frameobj = driver.findElement(By.xpath("//frame[@name='login_page']"));
driver.switchTo().frame(frameobj);
driver.findElement(By.xpath("//input[@name='fldLoginUserId']")).sendKeys("40058155");
driver.switchTo().defaultContent();
WebElement frameobj2 = driver.findElement(By.xpath("//frame[@name='footer']"));
driver.switchTo().frame(frameobj2);
driver.findElement(By.xpath("//a[text()='Terms and Conditions']")).click();
```

4.5 Switch to Alert:

- ✓ Alerts are the pop ups on the webpage, which cannot be identified by the firepath spy tool.
- ✓ To handle these, we need to switch to the alert and perform desired actions on it.
- ✓ At a time a webpage supports only one alert.
- ✓ The following are the steps to handle alerts on a webpage.
  - ❖ Switch to the alert
  - ❖ Get the text of the alert
  - ❖ Based on text accept or dismiss the alert.
  - ❖ After the alert is closed, the driver control will automatically switch back to the parent page.

Example:

```
Alert a1 = driver.switchTo().alert();
String text = a1.getText();
System.out.println(text);
if (text.contains("Customer ID  cannot be left blank"))
{
        System.out.println("About to accept Alert");
        a1.accept();
}
else
{
        System.out.println("About to dismiss Alert");
        a1.dismiss();
}
```

5. Cross Browser:

Firefox Browser:

```
WebDriver driver = new FirefoxDriver();
driver.get("http://testingmasters.com/hrm");
```

Chrome Browser:

Steps to download chrome driver:

- ✓ Navigate to *http://www.seleniumhq.org/download/*
  Google: selenium downloads
- ✓ In Third Party Browser Drivers Section, Click on the version name of Google chrome driver.

- ✓ Click on chromedriver_win32.zip to download the chrome driver
- ✓ Unzip the downloaded file to extract .exe file.

Use below code in the PSVM Method:

```
public static void main(String[] args)
{
        System.setProperty("webdriver.chrome.driver", "C: \\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.get("http://www.testingmasters.com/hrm");
}
```

IE Driver:

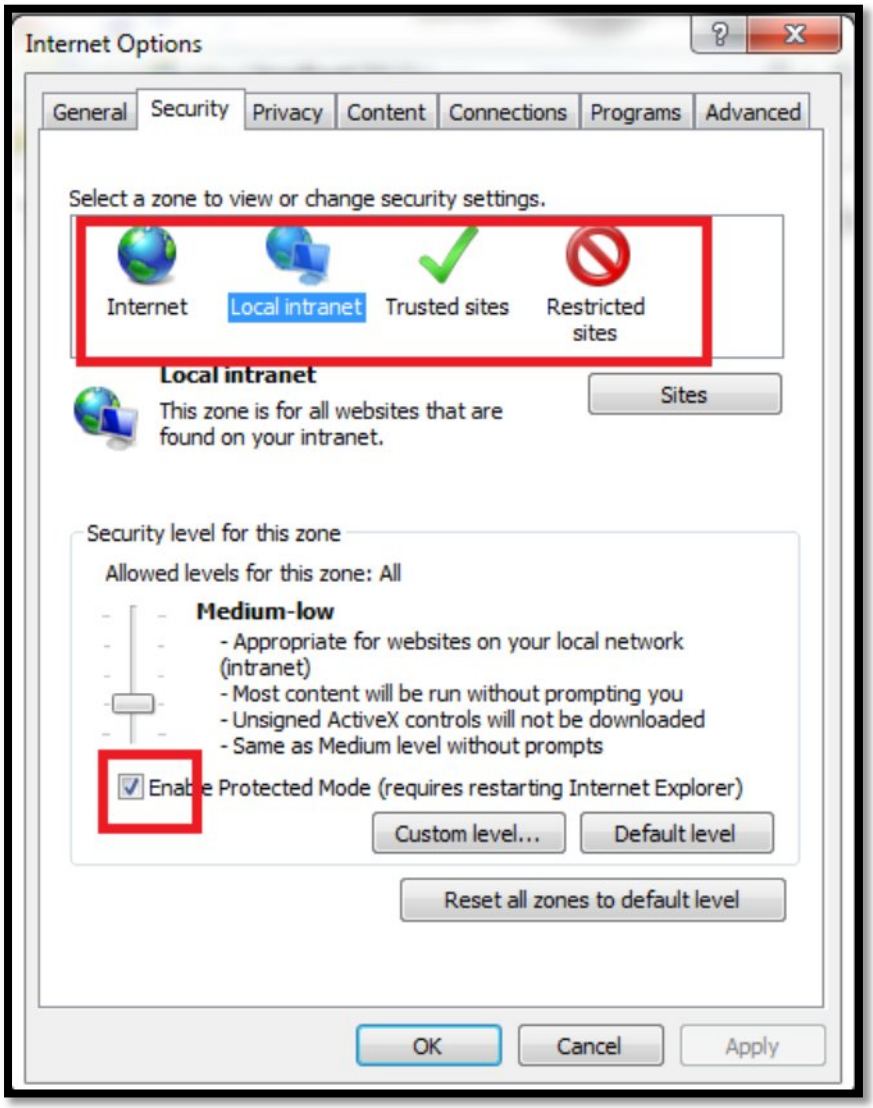
Steps to download IE driver:

- ✓ Navigate to *http://www.seleniumhq.org/download/*
  Google: selenium downloads
- ✓ Download the respective IE Driver server based on the system type
  [Right Click on my computer Properties System Type]
- ✓ Click on this link 64 bit Windows IE to download driver server.
- ✓ Unzip the zip file to extract .exe file.
  Use the below code in PSVM to work with IE Browser:

```
System.setProperty("webdriver.ie.driver", "IEDriverServer.exe");
WebDriver driver = new InternetExplorerDriver();
driver.get("http://testingmasters.com/hrm");
```

The following are restrictions while working with IE Browser:

- ✓ Protection mode should be same for all the zones in security settings.



- ✓ Zoom level should be set to 100%.

Desired Capabilities:

- ✓ Desired capabilities are created to introduce some settings, into the browser before launching.
- ✓ The following is the code to ignore the security settings and zoom level settings before launching IE Browser.

```
DesiredCapabilities capabilities = DesiredCapabilities.internetExplorer();
capabilities.setCapability(InternetExplorerDriver.IGNORE_ZOOM_SETTING, true);
capabilities.setCapability(InternetExplorerDriver.INTRODUCE_FLAKINESS_BY_IGNORING_SECURITY_DOMAINS, true);
System.setProperty("webdriver.ie.driver", "C: \\IEDriverServer.exe");
WebDriver driver = new InternetExplorerDriver(capabilities);
driver.get("http://www.testingmasters.com/hrm");
```

6. Synchronization:

- ✓ Synchronization is the Time Interface between the script and web page.
- ✓ The following are the synchronization statements used in selenium
  - ❖ Static Wait
  - ❖ Implicit wait
  - ❖ Explicit wait

Static wait:

- ✓ In this wait, the script will halt the execution for the specified time duration, independent of the web page behavior.
- ✓ The script will wait for the same time, even if you run it for any number of times.
- ✓ Thread.sleep is the method used for the static wait.
  Example:
  Thread.sleep(2000) --- For making the script wait for 3seconds
- ✓ Generally static wait is not preferred in the real time if the wait is more than 1-2seconds.
  Note:
  Static wait is not preferred for longer wait durations because, the execution will continue to halt even if the element is existing, before the specified time.
  Example:
  If static wait is specified for 10Sec and even if the desired element is loaded in 2sec, then also the script will wait for 10seconds, thus wasting remaining 8seconds of time.
- ✓ Static wait is applicable only for the current statement in selenium code.

Implicit Wait:

- ✓ In implicit wait, the existence of the webelement is verified for every 0.5 seconds
- ✓ At any point, if the webelement exists, execution will continue with the normal flow.
- ✓ If the wait time has reached the max specified time and still element is not existing, then it will throw an exception.
- ✓ Implicit wait will be applicable throughout the driver instance.
- ✓ In real time, Most of the cases we will use implicit wait.

Example:

```
WebDriver driver=new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

Note:

If Once implicit wait is defined, the algorithm of implicit wait will be applicable for all driver.findElement statements.

- ✓ if we declare implicit wait once that is enough for the whole script.

Explicit Wait:

- ✓ Explicit wait is same as implicit wait, but only difference is that explicit wait is applicable for the specified webelement.
- ✓ In General explicit wait is not much preferred.
- ✓ Explicit wait is suggest able only for those webelements,
  - ❖ which can take some abnormally long times to load the element
  - ❖ check for the conditions of a webelement.[Wait until the element is enabled]

Example:

```
WebDriverWait w1 = new WebDriverWait(driver, 40);
WebElement ele =
 w1.until(ExpectedConditions.presenceOfElementLocated(By.id("oway")));
ele.click();
```

7. Advanced User Actions:

* Mouse hover
* Right Click
* Drag N Drop
* Scroll down page
* Scroll down to webelement

Mouse Hover:

- ✓ Create object for action class.
- ✓ Use movetoelement method.
- ✓ Use .build() and .perform()

Example:

```
Actions a1 = new Actions(driver);
WebElement mainMenu = driver.findElement(By.xpath("//span[text()='Men']"));
a1.moveToElement(mainMenu).build().perform();
```

Right Click:

- ✓ Create object for Actions Object
- ✓ Use contect click method.
- ✓ use .build() and .perform()

Example:

```
WebDriver driver=new FirefoxDriver();
driver.get("https://talentzing.com/");
driver.manage().window().maximize();
driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
```

Drag n Drop:

- ✓ Create actions object
- ✓ use movetoelement and dragAndDrop Methods
- ✓ use .build() and .perform()

Example:

```
WebElement drag = driver.findElement(By.xpath("//div[@id=' draggable']"));
WebElement drop = driver.findElement(By.xpath("//div[@id=' droppable']"));
Actions action = new Actions(driver);
action.moveToElement(drag).dragAndDrop(drag, drop).build().perform();
```

Scroll Page:

- ✓ Create Object for JavaScriptExecutor by typecasting driver instance.
- ✓ use scollby method
- ✓ Scroll Right x +
- ✓ Scroll down y +
- ✓ Scroll Upy-
- ✓ Scroll left x-

Example:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
 js.executeScript("window.scrollBy(0,500)");
```

Scroll to WebElement:

- ✓ Create Object for JavaScript Executor
- ✓ Use scrollintoView Method

Example:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
 WebElement element = driver.findElement(By.xpath("//abbr[text()='01:44']"));
 js.executeScript("arguments[0].scrollIntoView();", element);
```

8. Excel:

- ✓ Download apache POI jar files.
- ✓ Configure the jar files to the eclipse project
  [Right Click on Project Build Path Configure Build Path Libraries Add External Jars Add the jar files]
- ✓ Create the objects Based on the Object Hierarchy Defined.
- ✓ To Create an object for the file instance, use below code

```
File f1 = new File("E:\\Tmasters\\Seleniu\\MercuryData.xlsx");
```

- ✓ Creating File Input Stream

```
FileInputStream fis = new FileInputStream(f1);
```

- ✓ Work book/WorkSheet objects has to be created based on the file extention type.

	.xls	.xlsx
WorkBook	HSSFWorkBook	XSSFWorkBook
WorkSheet	HSSFSheet	XSSFSheet

Example:

```
XSSFWorkbook wb1 = new XSSFWorkbook(fis);
XSSFSheet ws1 = wb1.getSheet("Sheet1");
```

- ✓ Cell Object can be created by using the row and column numbers.
  Syntax:
  ```
  Cell ObjectName = SheetObject.getRow(RowNumber).getCell(ColumnNumber);
  ```
  Example:
  ```
  Cell c1 = ws1.getRow(2).getCell(0);
  ```
- ✓ Read the data from the Cell
  ```
  System.out.println(c1.getStringCellValue());
  ```
  Note:
  - ❖ Index of a row will start from 0.
  - ❖ Index of a cell will start from 0.

Code to read Data Fom Excel:

```
File f1 = new File("C:\\Users\\balu\\Downloads\\MercuryData.xlsx");
FileInputStream fis = new FileInputStream(f1);
XSSFWorkbook wb1 = new XSSFWorkbook(fis);

XSSFSheet ws1 = wb1.getSheet("TestData");
Cell c1 = ws1.getRow(2).getCell(1);
System.out.println(c1.getStringCellValue());

wb1.close();
fis.close();
```

RowCount:

- ✓ To get the number of rows in the excel we can use
  ```
  System.out.println(ws1.getLastRowNum());
  ```

CellCount:

- ✓ To get cell count in the excel, we can use
  ```
  System.out.println(ws1.getRow(4).getLastCellNum());
  ```
  Here, 4 is the row number.

NOTE:

- ✓ getLastRowNum will return the last index of a row
- ✓ Whereas getLastCellNum will return last index of a cell + 1.

Code to Print the Value of all the cells in Row 1:

```
int columncount = ws1.getRow(1).getLastCellNum();
for(int cn = 0;cn<columncount;cn++)
{
        Cell c1 = ws1.getRow(1).getCell(cn);
        System.out.println(c1.getStringCellValue());
}
```

Code to Print all the values of a Sheet of excel:

```
File f1 = new File("E:\\Tmasters\\Selenium\\MercuryData.xlsx");
FileInputStream fis= new FileInputStream(f1);
XSSFWorkbook wb1 = new XSSFWorkbook(fis);

XSSFSheet ws1 = wb1.getSheet("Sheet1");
int rowcount = ws1.getLastRowNum();
for(int i = 0;i<=rowcount;i++)
{
        int colcount = ws1.getRow(i).getLastCellNum();
        for(int k =0;k<colcount;k++)
        {
                Cell c1 = ws1.getRow(i).getCell(k);
                System.out.println("Row:="+ i + ";" + "Column:=" + k + " : "+
                c1.getStringCellValue());
        }
}
wb1.close();
fis.close();
```

Code to close the inputStream and workbook:

```
wb1.close();
fis.close();
```

Code to Write Data to a cell of Excel:

```
File f1 = new File("E:\\MercuryData.xlsx");
FileInputStream fis = new FileInputStream(f1);
XSSFWorkbook wb1 = new XSSFWorkbook(fis);
XSSFSheet ws1 = wb1.getSheet("Sheet1");
Cell c1 = ws1.getRow(1).getCell(1);
if (c1==null)
{
        c1 = ws1.getRow(1).createCell(1);
}
c1.setCellValue("NewPassword");
fis.close();
FileOutputStream fos = new FileOutputStream(f1);
wb1.write(fos);
fos.close();
wb1.close();
```

10. Selenium RC:

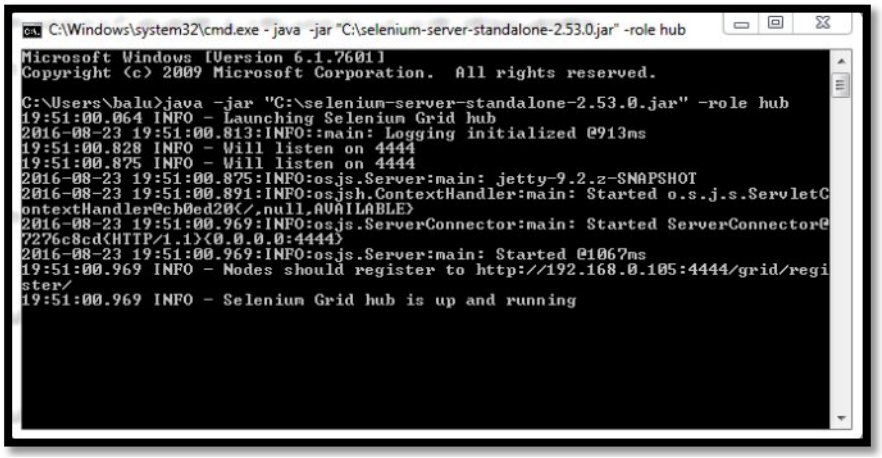
- ✓ Selenium RC stands for selenium Remote control.
- ✓ The following is the architecture of selenium RC.
- ✓ The following are the limitations of selenium RC:
  - * It uses selense commands which are java script commands
  - *To decrypt these commands, selenium uses RC Server.
  - *And to launch the server, some of the ports have to be opened.
- ✓ Because of these limitations selenium webdriver is introduced.

11. Selenium Grid:

- ✓ Selenium Grid can be used for remote execution/Parallel Execution of the testcases.
- ✓ The following are the major components in the selenium grid.
  - * Hub
  - * Node
- ✓ The following are the key things that are involved in Selenium Grid setup and execution
  - * Creation of Hub
  - * Creation of Node and register to the Hub
  - * Verify the registration to the Hub
  - * Trigger execution in a node from the hub.

Creation of Hub:

- ➢ Open command prompt
- ➢ Type the following command by specifying the selenium server jar file path.
  java -jar "C:\selenium-server-standalone-2.53.0.jar" -role hub
- ➢ After clicking on enter the following message will be displayed

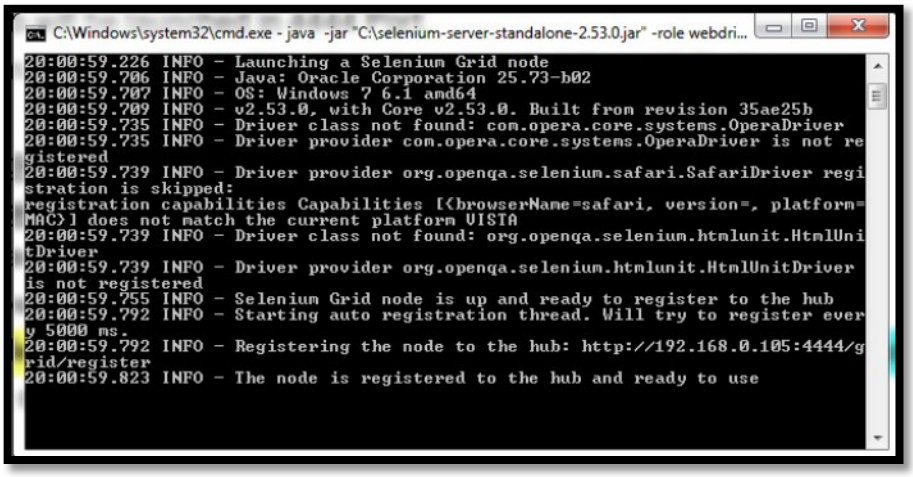


- ➢ By default hub will be launched in 4444 Port.
- ➢ The same can be viewed from "http://localhost:4444/grid/console" URL

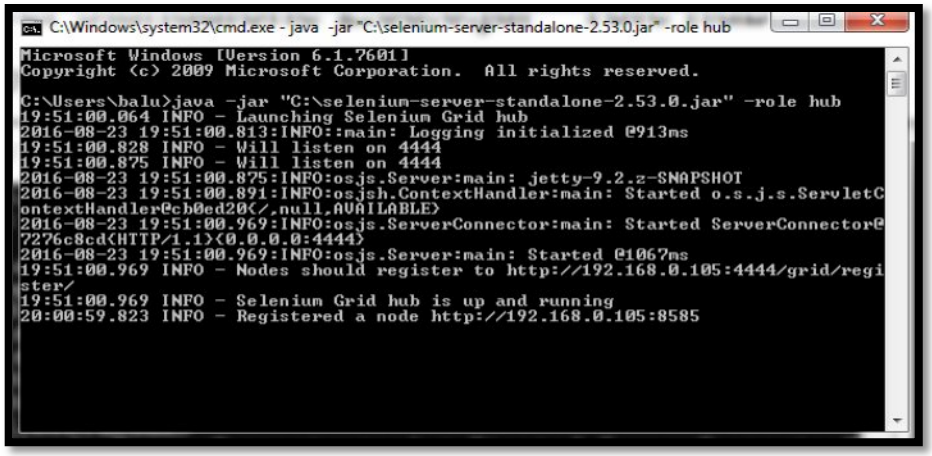
Creation of Node and register to the Hub:

1) Register Local Machine Firefox Node:

- ✓ Open command prompt
- ✓ Type the following command by specifying the selenium server jar file path.
  java -jar "C:\selenium-server-standalone-2.53.0.jar" -role webdriver -hub http://192.168.0.121:4444/grid/register -port 8585
- ✓ After Clicking on enter the following message will be displayed

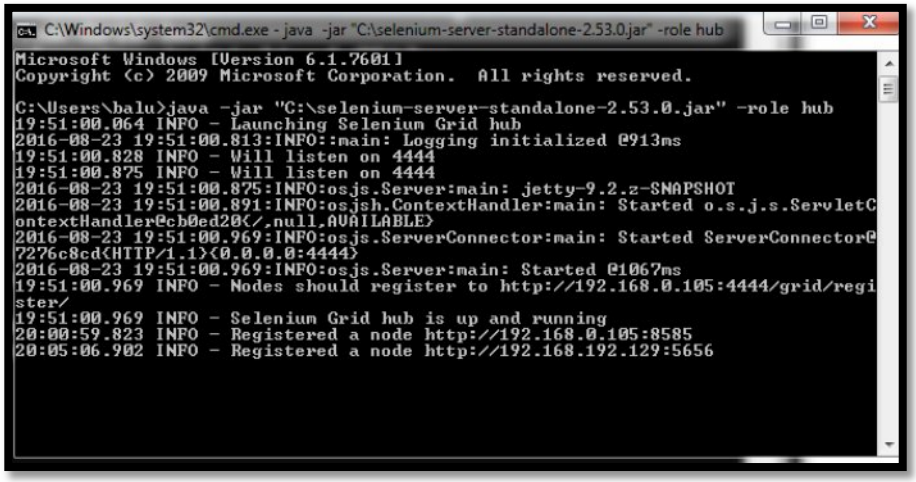


- ✓ The same can be viewed in the



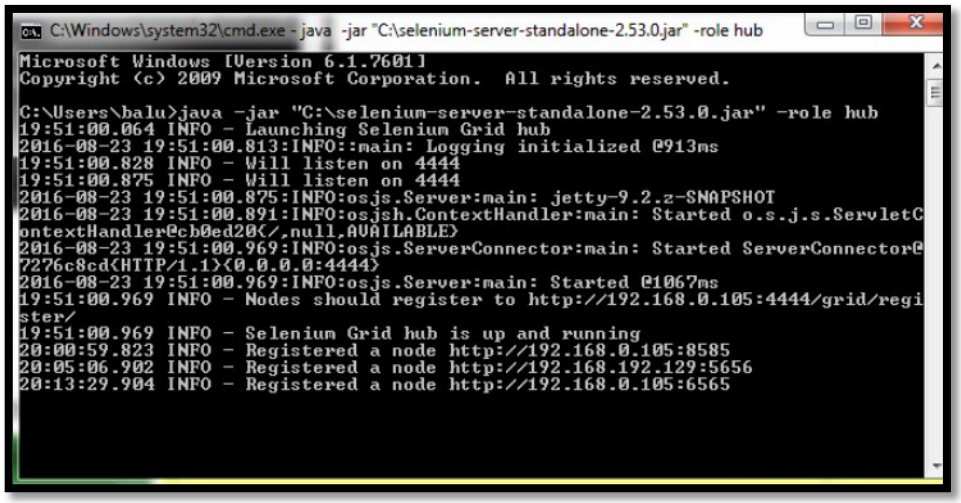
2) Register Remote Firefox Node:

- ✓ Open command prompt
- ✓ Type the following command by specifying the selenium server jar file path.
  java -jar "C:\selenium-server-standalone-2.53.0.jar" -role webdriver -hub http://192.168.0.105:4444/grid/register -port 8585
- ✓ The same can be viewed in the



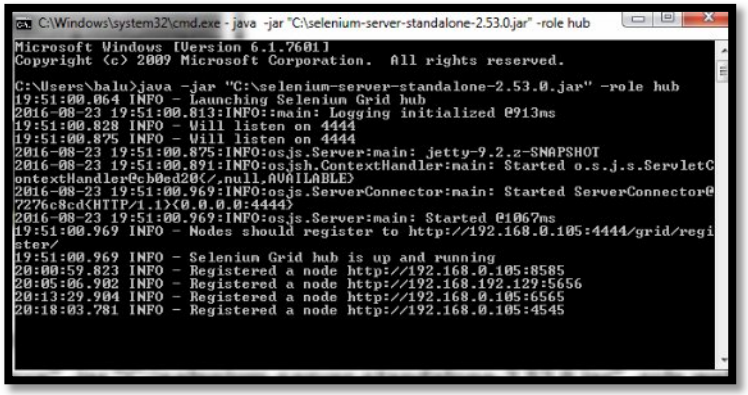
3) Register Local Chrome Node:

- ✓ Open command prompt
- ✓ Type the following command by specifying the selenium server jar file path.
  java -Dwebdriver.chrome.driver="C:\chromedriver.exe"
  -jar "C:\selenium-server-standalone-2.53.0.jar" -role webdriver
  -hub http://192.168.0.105:4444/grid/register
  -port 6565
- ✓ The same can be viewed in the



4) Register Local IE Node:

- ✓ Open command prompt
- ✓ Type the following command by specifying the selenium server jar file path.
  java -Dwebdriver.ie.driver="C:\IEDriverServer.exe"
  -jar "C:\selenium-server-standalone-2.53.0.jar" -role webdriver
  -hub http://192.168.0.105:4444/grid/register
  -port 4545



Configure hub with selenium code:

The following is code for creating a werbdriver object in a node

```
DesiredCapabilities capability = DesiredCapabilities.firefox();
capability.setBrowserName("firefox");
capability.setPlatform(Platform.WINDOWS);
WebDriver driver = new RemoteWebDriver(new URL("http://
192.168.0.105:5656/wd/hub"),capability);
```

Note:

Desired capabilities are used to configure the settings of the browser instance.

12. AutoIT:

Auto It V3 is a freeware tool which is used for automating anything in Windows environment. Auto It script is written in BASIC language. It can simulate any combination of keystrokes, mouse movement and window/control manipulation.

While doing automation through Selenium or through any other tool for that matter, we all encounter a common problem, windows pop-ups. As Selenium is confined to automating browsers, desktop window is out of scope. Web applications sometimes need to interact with the desktops to perform things like file downloads and uploads. There are tools available for automating these sorts of workflow such as *AutoIt* .

We can upload or download the files or images by transferring our control from Selenium WebDriver to AutoIt. We need to explicitly call the AutoIt script from our program.

Steps to use AutoIT:

Find the properties using the AutoIT finder tool

- Drag the Finder tool and drop it on the desired windows object.
- The Properties of the element such as Title, Class and instance will be displayed.
- ControlID = Class + Instance

Example:

Class = Edit
Instance = 1
Then, ControlID = Edit1.

- Use the title and Control ID in the scripting.
- The following are the commands used for AutoIT Scripting

*ControlFocus -- focus on the object
*ControlClick -- click on the object
*ControlSetText -- enter text
*Sleep -- wait in milli seconds

Example:

```
Sleep(3000);
ControlFocus("File Upload","","Edit1");
ControlClick("File Upload","","Edit1");
ControlSetText("File  Upload","","Edit1","C:\abcd.xlsx");
Sleep(3000);
ControlClick("File Upload","","Button1");
```

- Save the code with .au3 extention.
- RightClick on the .au3 extention fileCompile the script
- .Exe file will be generaed in the same path.
- Trigger this exe file to run the AutoIT Script.

Code to trigger .exe file from Java:

```
Runtime.getRuntime().exec("C:\\Users\\Tmasters\\Desktop\\Browse.exe");
```

Selenium Code Example:

```
WebElement element = driver.findElement(By.xpath("//input[contains(@name,'flUpdResume')]"));
element.click();
Thread.sleep(2000);
Runtime.getRuntime().exec("C:\\Users\\Tmasters\\Desktop\\Browse.exe");
```

13. Robot Keys:

- ✓ Robot Keys are used to generate keyboard actions from java.
- ✓ For this, first we need to create an object for Robot class.
- ✓ Then, use KeyPress or KeyRelease methods based on the need.
- ✓ The following is the code to copy a specific text to clipboard

```
String filepath = "E:\\Tmasters\\Selenium\\TmBatch19_Sept_7.30AM\\Collections.docx";
StringSelection selection = new StringSelection(filepath);
Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
clipboard.setContents(selection,selection);
```

Code to browse file using ROBOT Keys in Java:

```
driver.findElement(By.xpath("//input[@class='browsefile']")).click();
Thread.sleep(2000);
String filepath = "E\\Collections.docx";
StringSelection selection = new StringSelection(filepath);
 Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
clipboard.setContents(selection,selection);
```

```
Robot robo = new Robot();
robo.keyPress(KeyEvent.VK_CONTROL);
robo.keyPress(KeyEvent.VK_V);
robo.keyRelease(KeyEvent.VK_V);
robo.keyRelease(KeyEvent.VK_CONTROL);
Thread.sleep(3000);
robo.keyPress(KeyEvent.VK_ENTER);
robo.keyRelease(KeyEvent.VK_ENTER);
```

14. TestNG

TestNG Stands for Test Next Generation Framework, it is software which provides some facilities through annotations which make the testing job easier as it is making some(testing) job more easy people generally call it as TestNG Framework.

Steps to Install TestNG:

- ✓ Open Eclipse
- ✓ Go to Help>>Install New Software
- ✓ Click on add, provide below details
  Name: TestNG
  Location: http://www.beust.com/eclipse
- ✓ Select the TestNG Checkbox and click on Next
- ✓ Accept license agreement and click on finish.

Annotation in TestNG:

- ✓ Annotations in testNG are used to control the flow of execution of program at run time.
- ✓ The following are the annotations that can be used in TestNG
  @Test
  @Beforetest
  @AfterTest
  @BeforeSuite
  @AfterSuite
  @BeforeClass
  @AfterClass
  @BeforeMethod
  @AfterMethod
  @DataProvider

Steps to create a First TestNG class:

- ✓ Right Click on package>>New>>Others>>Testng class
- ✓ Provide the name, Select the required annotations and click on finish.
- ✓ By Default a methods will be created with @Test Annotation.
- ✓ Write a code in the body of default annotation and click on run.
- ✓ we can also execute by right clicking on .java file>run as >> TestNG Suite/Test

Multiple Methods:

- ✓ If we have multiple methods under the same annotation, by defualt they will be executed in the alphabetical order of the method names.
- ✓ The Order of execution of the methods is independent of the Order of writing the methods on the code.

```
public class Test1
{
  @Test
  public void Method1()
  {
      System.out.println("Executing Method1");
  }
  @Test
  public void Method3()
  {
      System.out.println("Executing Method3");
  }
  @Test
  public void Method2()
  {
      System.out.println("Executing Method2");
  }
}
```

Output:
Executing Method1
Executing Method2
Executing Method3

Setting Priorities for TestNG:

- ✓ By default TestNG Methods will be executed in the alpahabetical order of the method names.
- ✓ If we want to redefine the order of execution of the methods, then we have to set the priorities for each method.

Example:

```
@Test(priority = 1)
 public void Login()
 {
          System.out.println("Method to Login into application");
 }
```

Note:

- ❖ Priorities can also take nagative values.
- ❖ Priorities can be any integer no necessarily consecutive integers.
- ❖ If more than one test is having same priority, then alphabetical oder of the methods are considered.

Example:

```
public class Test1
{
 @Test(priority = -100)
 public void Login()
 {
  System.out.println("Method to Login into application");
  }
@Test(priority = 2)
 public void AddSkillSetDetails()
 {
          System.out.println("Method to AddSkillSetDetails");
 }
@Test(priority = 10000)
```

```
public void Logout()
{
        System.out.println("Method to Logout");
}
@Test(priority = 3)
public void AddEducationDetails()
{
        System.out.println("Method to AddAddEducationDetails");
}
}
```

Output:

```
Method to Login into application
Method to AddSkillSetDetails
Method to Logout
Method to AddAddEducationDetails
```

@BEFORETEST,@AFTERTEST:

- ✓ The method under @BeforeTest annotation, will be executing before first method under @Test annotation.
- ✓ The Method under @AfterTest annotation will be executed after the last method under @Test annotation.

Example:

```
public class Test1
{
@Test(priority = -100)
public void Login()
{
        System.out.println("Method to Login into application");
}
@Test(priority = 2)
public void AddSkillSetDetails()
 {
        System.out.println("Method to AddSkillSetDetails");
}
@Test(priority = 10000)
public void Logout()
{
        System.out.println("Method to Logout");
}
@Test(priority = 3)
public void AddEducationDetails()
{
        System.out.println("Method to AddAddEducationDetails");
}
@BeforeTest
public void LaunchBrowser()
{
        System.out.println("Method to LaunchBrowser");
}
@AfterTest
public void CloseBrowser()
```

```
{
        System.out.println("Method to CloseBrowser");
 }
}
```

Output:

```
Method to LaunchBrowser
Method to Login into application
Method to AddSkillSetDetails
Method to AddAddEducationDetails
Method to Logout
Method to CloseBrowser
```

Steps to Create A XML File for Executing TestNg Clasess:

- ✓ Right Click on the package>>Go to TestNG>>Convert to TestNG>>Specify Details>>Click on Finish.
- ✓ A XML File will be created at the project location.
- ✓ double click on XML file to edit the details in XML file.
- ✓ Right Click on XMl File>>Run as>>TestNG Suite to run the test cases inside a package.
- ✓ The following is an example XML File of executing the multiple test cases inside a package

```
<suite name="Suite">
 <test name="Test">
  <classes>
   <class name="pc1.TestCase1"/>
   <class name="pc1.TestCase2"/>
  </classes>
 </test>
</suite>
```

- ✓ The following is an example XML File of executing all the test cases inside a package

```
<suite name="ExecutionSuite">
 <test name="Execution1">
  <packages>
   <package name="pc1.*" />
  </packages>
 </test>
</suite>
```

Grouping The Test Cases in TestNG:

- ✓ To add a specific testNg Method to a group , we use mention them as below

```
@Test(priority = 3,groups="group1")
public void Logout()
{
        System.out.println("Logout");
}
```

- ✓ For adding the same method under multiple groups, we can mention as below

```
@Test(priority = 2,groups={"group1","group2"})
public void CreateAdmin()
{
 System.out.println("CreateAdmin");
}
```

- ✓ The following is the XML File of executing the testcases based on the group names.

```
<suite name="ExecutionSuite">
```

```
<test name="Execution1">
        <groups>
                <run>
                        <include name="group1" />
                        <exclude name="group2" />
                </run>
        </groups>
 <packages>
  <package name="pc1.*" />
 </packages>
 </test>
</suite>
```

Note:

With the above XML File,

>> all the methods which belong to group2 are excluded for execution.

>> and all the methods which belongs to group1 are included for execution

First the Methods will be excluded, Next the remaining methods will be filtered for inclusive criteria.

Parameterization In TestNG from XML:

- ✓ The below is the format for writing a TestNG Method which can accept parameters from XML

```
@Test
@Parameters("UserID")
 public void parameterTest(String str1)
 {
 System.out.println("Parameterized value is : " + str1);
 }
```

- ✓ The following is the XMl Format for passing the value.

```
<suite name="ExecutionSuite">
 <test name="Execution1">
   <parameter name="UserID" value="user01"/>
   <classes>
     <class name="TestNG.DataProviderClass" />
   </classes>
 </test>
</suite>
```

Using Data Providers in TestNG:

- ✓ DataProviders in TestNg can be used to execute a test case with multiple set of test data.
- ✓ The following is an example for executing a method with data provider.

```
public class DataProviderExample
{
//This test method declares that its data should be supplied by the Data Provider
        // "getdata" is the function name which is passing the data
    // Number of columns should match the number of input parameters
        @Test(dataProvider="getData")
        public void setData(String username, String password)
```

```
        {
                System.out.println("you have provided username as::"+username);
                System.out.println("you have provided password as::"+password);
        }

        @DataProvider
        public Object[][] getData()
        {
        //Rows - Number of times your test has to be repeated.
        //Columns - Number of parameters in test data.
        Object[][] data = new Object[3][2];

        // 1st row
        data[0][0] ="sampleuser1";
        data[0][1] = "abcdef";

        // 2nd row
        data[1][0] ="testuser2";
        data[1][1] = "zxcvb";

        // 3rd row
        data[2][0] ="guestuser3";
        data[2][1] = "pass123";

        return data;
        }
}
```

Parallel Execution In testNg:

- ✓ TO Execute testcases Paralally in TestNG, we can use below XML Format

```
<suite name="ExecutionSuite" parallel="methods" thread-count="3">
 <test name="Execution1">
  <packages>
   <package name="parallelExecution.*" />
  </packages>
 </test>
</suite>
```

- ✓ Thread Count Specifies the number of tests that can be executed at a time.

15. Maven:

Maven is a project management and comprehension tool. Maven provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.

In case of multiple development teams environment, Maven can set-up the way to work as per standards in a very short time. As most of the project setups are simple and reusable, Maven makes life of developer easy while creating reports, checks, build and testing automation setups.

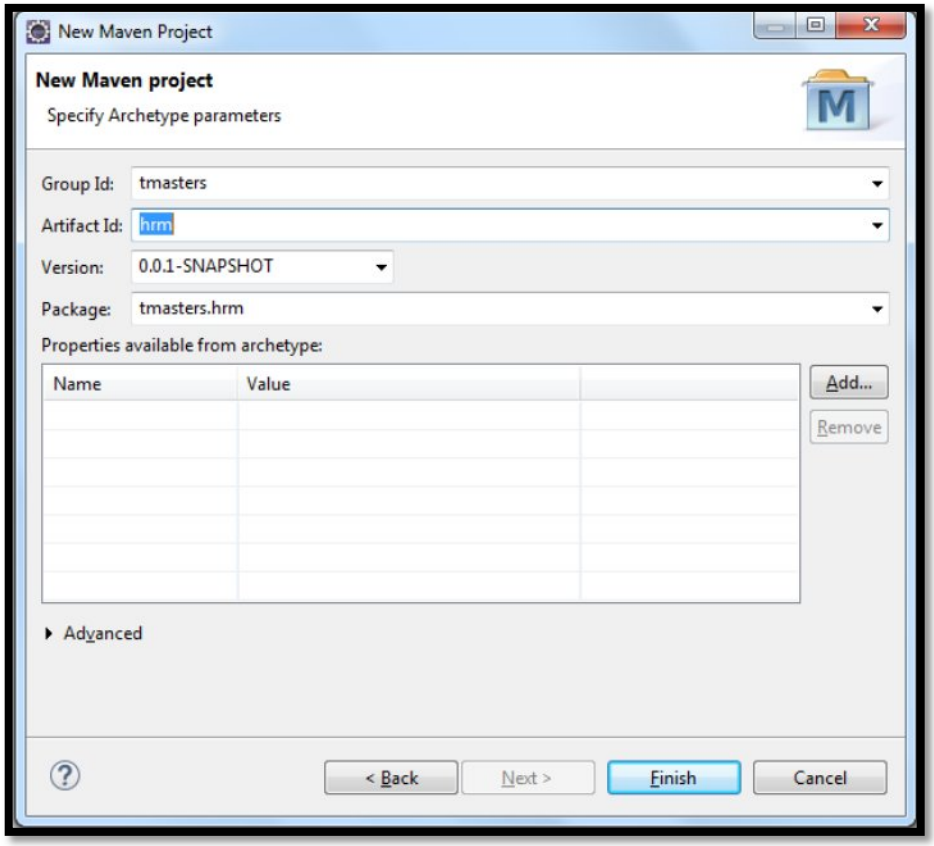
Maven primary goal is to provide

- A comprehensive model for projects which is reusable, maintainable, and easier to comprehend.
- Plug-in or tools that interact with this declarative model.

Maven project structure and contents are declared in an xml file, pom.xml referred as Project Object Model (POM), which is the fundamental unit of the entire Maven system

Steps to create Maven Project:

- ✓ Open eclipse[preferably version above helper]
- ✓ Go to file>>New>>Other>>Maven Project>>Next>>Next>>Next
- ✓ Specify GrouID,ArtifactID.
- ✓ Generally GroupId: Company Name
  ArtifactID: ProjectName
- ✓ Package name will be auto populated as GroupID.ArtifactID



- ✓ On clicking finish, Maven Project will be created along with pom.XML file.
- ✓ IN this XML file, we need to add the dependencies[Unique ID] of the respective jar files.
- ✓ By default junit dependency will be added.
- ✓ Remove junit dependency and remove the unnecessary files.
- ✓ Now, get the required Unique ID [dependency] of the required jar file from https://mvnrepository.com

- ✓ Paste this dependency in POM.xml file and click on save.
- ✓ Now, the respective jar files will be downloaded from maven repository and associated with the current maven project.
- ✓ In This way the jar files will be associated with the project automatically based on the specified dependency.

16. Framework:

- ✓ A framework is a set of rules or guidelines that are to be followed for easy development and maintenance of the test scripts.

  Types of Frameworks:

  * Linear Framework
  * Data Driven Framework
  * Keyword Driven Framework
  * Modular Framework
  * Hybrid Driven Framework.

Linear Framework:

- ✓ The following are the disadvantages of using the linear coding

  *No Log file/Reports
  *Multiple test case execution is not possible
  *Error handling is not easy
  *Maintenance of test data is difficult
  *Maintenance of the test scripts is difficult.
  *Reusability of the code is not achieved.
  *To overcome these limitations the other framework types are introduced.

Data Driven framework:

- ✓ In data driven framework the testdata is placed in some external sources such as Excel Files/Databases
- ✓ As the execution is driven from the data taken from external sources this is named as data driven framework.
- ✓ The following are the key features of the data driven framework

  * TestData should be placed in external sources
  * Provision for easy maintenance of the testdata.
  * Provision for executing a testcase with multiple sets of testdata.

TestCaseName	Iteration			
TC01_VerifyLogin	1	URL:=http://www.testingmas	UserName:=user01	Password:=PASS1234
TC01_VerifyLogin	2	URL:=http://www.testingmas	UserName:=user02	Password:=PASS1235
TC02_AddEmergencyContactsAndVerify	1	URL:=http://www.testingmas	UserName:=user01	Password:=pass1234
TC03_AddDependenciesAndVerify	1	URL:=http://www.testingmas	UserName:=user01	Password:=PASS1234
TC04_ApplyLeave	1	URL:=http://www.testingmas	UserName:=user01	Password:=pass1234

Keyword Driven Framework:

- ✓ In this framework, some generic methods/classes will be created which can be accessed across all the modules/projects.
- ✓ These keywords are designed considering the overall usage of the methods of a webdriver.
- ✓ The keywords can be of two types
  * Web General Keywords
  * Application General Keywords
- ✓ These keywords are written along with error handling and acknowledgements.
- ✓ Using these keywords, will ease the process of the development of test scripts.

Modular Driven Framework:

- ✓ In Modular driven framework, the common test steps across all the test cases in a module are identified.
- ✓ And methods are created for these common steps which can resued across all the testcripts of that respective module.

- ✓ By using these reusable methods, the maintenance of the test scripts will be abetted easy when the properties of the objects got changed.

Hybrid Driven Framework:

- ✓ Hybrid driven framework can be combination of two or more types aof the above three types of frameworks.
- ✓ Tmasters framework is the combination of data driven, hybrid driven and keyword driven frameworks.
- ✓ This framework has following components.
  Components of framework:
  * Trigger.bat file
  * RunManager
  * TestData
  * Driver Script
  * Generic Resources
  * TestCase Definitions
  * Jar Files
  * Results

Trigger.Bat:

- ✓ This file is used to trigger the executions of the testcases.
- ✓ Trigger.bat file will intern invokes the driver script/Test Runner script.
- ✓ Test Runner Script will Take care of the further executions.
- ✓ This is framework specific file

Run Manager:

- ✓ This is an excel file in which all the testcase information is stored along with execution flag
  The following is the template of the Run Manager file

TestCaseName	TestCaseDescription	Browser	Execute
TC01_VerifyLogin	Verify Login of HRM Portal	firefox	True
TC02_AddEmergencyContactsAndVerify	Add Emergency contact details and verify	firefox	False

- ✓ Here the test engineer can select the testcases that are to be executed in the current Run.
- ✓ So, for the test script development, the first task will be to mention the testcase in the runmanager.xls file.
- ✓ This is a project specific file

TestData:

- ✓ In this folder we have to specify the testdata that is required for the execution of the testcase
- ✓ The TestData is stored against the testcase names in these files
- ✓ The following is the template of specifying the testdata

TestCaseName		
TC01_VerifyLogin	URL:=http://www.testingmasters.com/hrm/symfony/web/index.php/auth/login	UserName:=user02
TC02_AddEmergencyContactsAndVerify	URL:=http://www.testingmasters.com/hrm/symfony/web/index.php/auth/login	UserName:=user01

- ✓ The Testdata is stored in the format of FieldName:=TestData For That Field
  Example:
  UserName:=user02
  Name:=Srinivas

Driver Script/TestRunner Script:

- ✓ This is the core script of the framework which drives the complete execution.
- ✓ Test Runner Contains the public static void main method.
- ✓ Test Runner script opens the Runmanager file and checks, what are the testcases that are set for execution.
- ✓ All the testcases that are selected are executed one by one sequentially.
- ✓ This is framework specific script.

JarFiles:

- ✓ In This Folder all the required jar files for the framework are placed.
- ✓ These jar files includes POI, Selenium, Extent Report files.
- ✓ So, for setting a project, first we need to Add these jar files to the project in the eclipse.
- ✓ This is framework specific component.

TestCase Definitions:

- ✓ In this, we will define the actual test scripts code.
- ✓ We will create a method for every testcase, inside test.testcases package.
- ✓ The testcase name should be same in RunManager, TestData Sheet, method Name in test. testcases package.
- ✓ While writing the testcases, we will use the necessary methods which are already defined in weblibrary and userlibrary classes.
- ✓ This is a project specific folder

Results:

- ✓ In this folder, results will be generated for every run.
- ✓ For every run, a folder is created with a date and time stamp.
- ✓ Inside the folder the respective screenshots and results summary.html is stored.
- ✓ This will be a project specific folder

Generic Resources:

- ✓ This contains the reusable methods for working with excel sheets, Report generation, Working with Web Pages etc..
- ✓ This is generally a framework specific folder.
- ✓ The methods or classes in this folder can be used by the driver script or testcase definitions.
- ✓ These are specified in test.resources.generic package.
- ✓ These methods are defined in the WebLibrary of Generic resources.

WebLibrary:

- ✓ In this library all the methods that are required for the interactions with the webpage are specified.
- ✓ These methods can be used while developing a testcase for performing operations on the webelements.
- ✓ The following are the sample methods in WebLibrary
  - *SetText
  - *ClickElement
  - *SetTextAndEscape
  - *Exist
  - *SelectOPtionByText
  - *SelectOPtionByValue
  - *OpenUrl

- ✓ The methods in this library are generic, framework specific and can be useful in writing the testcase.

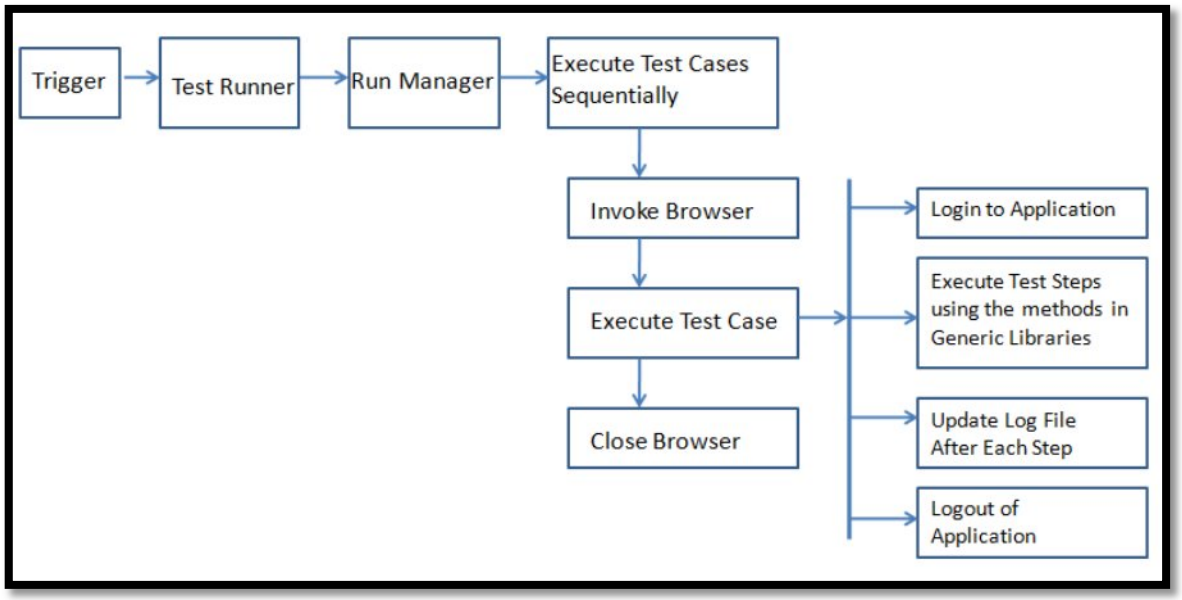
FrameWork Library:

- ✓ In this library, the methods that are required for the functioning of the framework are placed.
- ✓ Most of the methods present in this library are used internally by the framework.
- ✓ very rarely these methods will be used in the development of the testcases.
- ✓ The methods in this library are generic, framework specific and will be useful for framework functioning.

UserLibrary:

- ✓ In this library, the Test Engineer can add methods for the reusable steps across the testcases.
- ✓ Before/After writing the testcases, Test Engineer will check for the reusability components in the testcase.
- ✓ Will check if the method already exists for this functionality.
- ✓ If existing, TE can use that method in the testcase.
- ✓ Else, TE can add a method to the user library.
- ✓ The methods in this library are generic, project specific and can be useful in writing the testcase.

FrameWork Flow:



Java Basics:

Variable:

- ✓ a variable is a named memory space created at the runtime.

```
int x;
x = 5;
System.out.println(x);
```

- ✓ x is a variable here, for which the memory is created at runtime. D

DataType:

- ✓ DataType specifies the type of data that a variable can store.

Operators in java:

- ✓ Assignment operators: =
- ✓ Concatenation operator: +
- ✓ Arithmetic operators: + , - , * , /,%
- ✓ Comparison operator: <,>,<=,>=,==,!=
- ✓ Logical operator: &&,||,!
- ✓ Ternary operator: ?:

Conditional Statements:

- ✓ Conditional statements are used to make a decision
- ✓ In java we have two types of conditional statements.
  - ➔ If ...else
  - ➔ Switch... case

If--- Else:

- ✓ Whenever we want to execute one block of statements among two blocks then we can use If-Else.

Syntax:

```
if(condition)
{
        block of stmts1
}
else
{
        block of stmts2
```

```
}
```

- ✓ If the condition is true then block of stmts1 will be executed
- ✓ If the condition is false, then the block of stmts2 will be executed.

Switch...case:

- ✓ Whenever we want to execute a particular block of statements among many blocks then we will choose switch case statement.

Syntax:

```
switch(choice)
{
case "c1":
        block1
        break;
case "c2":
        block 2
        break;
----------
Default: ------ // not mandatory
}
```

Looping Statements:

- ✓ Looping statements are used for executing a certain block of statements repeatedly and continuously for some number of times.
- ✓ In java we have 3 types of looping statements
  - ➔ for loop
  - ➔ while loop
  - ➔ do...while loop

for loop:

- ✓ Whenever we clearly know how many no of times the block should be repeated then use for loop.

  syntax:

  for(initiation ;condition ;improvement)

  {

  ---------- block of stmts

  }

While loop:

- ✓ It is used for executing a block of statements repeatedly as long as the condition is being satisfied.

  Syntax:

  while(condition)

  {

  ------- block of stmts

  }

do- while loop:

- ✓ It is used for executing a block of statements repeatedly as long as the condition is being satisfied, but first time execution will be done without checking the condition.

  Syntax:
  do
  {
  ----------block of stmts

  } while(condition) ;

Arrays:

- ✓ An array is a collection of similar data types.
- ✓ Array is also known as static data structure because size of an array must be specified at the time of its declaration.
- ✓ Index of array starts from zero
- ✓ Index of array will end at length-1.

Example:

```
package coreJava;
public class P06_Arrays
{
	public static void main(String[] args)
	{
	int[] arr = new int[10];
	for(int i=0;i<10;i++)
	{
		System.out.println("Current Value in Array at index : "+arr[i]);
		arr[i]=i;
	}
	System.out.println("-------------------------------------------------");
	for(int i=0;i<10;i++)
	{
		System.out.println("Current Value in Array at index : "+arr[i]);
	}
	}
}
```

String Methods

Length:

- ✓ This method will return the number of characters in a string.

Example:

```
String str1 = "abcdef";
System.out.println(str1.length()); --6
```

CharAt:

- ✓ This method is used to get the character at the specified index of a string.

Example:

```
String str1 = "abcdef";
System.out.println(str1.charAt(0)); --a
System.out.println(str1.charAt(3)); --d
```

Input: Index

Output: Character at specified index

Note:

- ❖ For a string, the index will start from 0 and end at n-1[where n is the length of a string]

Program to Print each Character of a String:

```
String str1 = "abcdef";
for(int i = 0;i<str1.length();i++)
{
        System.out.println(str1.charAt(i));
}
```

Starts-With:

- ✓ This method verifies if a main string is starting with the specified set of characters.
  If Yes returns true
  else returns false.

  Example:

```
String str1 = "abcdefg";
System.out.println(str1.startsWith("abc"));  -- returns true
System.out.println(str1.startsWith("bcd"));  -- returns false
```

Ends-With:

- ✓ This method verifies if a main string is ending with the specified set of characters.
  If Yes returns true
  else returns false.

  Example:

```
String str1 = "abcdefg";
System.out.println(str1.endsWith("efg"));  -- returns true
System.out.println(str1.endsWith("xyzefg"));  -- returns false
```

Contains:

- ✓ This method verifies if a main string contains the specified set of characters independant of the position.
  If Yes returns true
  else returns false.

  Example:

```
String str1 = "abcdefg";
System.out.println(str1.contains("cde"));-- returns true
System.out.println(str1.contains("xyzefg"));  -- returns false
```

indexOf:

- ✓ This method will verify the existence of the substring in the main string,
  if exists, returns the first matching character position.
  else, returns -1.

  Input: Set of characters
  Output: Index Position

  Example:

```
String str1 = "abcdefg";
System.out.println(str1.indexOf("cde")); -- 2
System.out.println(str1.indexOf("x"));  --  -1
```

  Note:

  - ❖ IndexOf Method will consider only the first occurance of the sub string in the main string.
  - ❖ LastIndexOf Method will consider only the last occurance of the sub string in the main string.

sub String:

- ✓ This method is used to extract a sub string from the main string, based on the index positions specified.
  Inputs: Begin Index Position, End Index Position{Optional}
  Output: sub string from the main string
  Syntax:
  MainString.Substring(BeginIndex, EndIndex)
  Begin Index Character is Inclusive.
  End Index Character is Exclusive.
  Note:
  - ❖ If End Index is not specified, it will condier till the end character position.

Split:

- ✓ This method is used to extract sub strings from main string , based on the specified character{Delimiter}
- ✓ This method splits the main string and returns in the form of string array.

Syntax:

```
MainString.Split(Delimiter)
```

Example1:

```
String str1 = "AP;TG;TN;KA;DEL";
String []arrstr = str1.split(";");
for(int i=0;i<arrstr.length;i++)
{
        System.out.println(arrstr[i]);
}
```

Example2:

```
String str1 = "AP;TG;TN;KA;DEL";
String []arrstr = str1.split(";");
for(String ele:arrstr)
{
        System.out.println(ele);
}
```

Join:

- ✓ This method is used to join the contents of an array using the delimiter.
- ✓ This method joins the array values and returns result in the form of string.

Example:

```
String str2;
String []arrstr = {"AP","TG","DEL"};
str2 = String.join(",", arrstr);
System.out.println(str2);
```

Replace:

- ✓ This method is used to replace a set of old characters with a set of new characters

syntax:

```
String.Repalce(Old Set of Characters,New Set of Characters)
```

Example:

```
String str1="welcome to hyderabad";
System.out.println(str1.replace("hyderabad", "chennai"));
```

to UpperCase,toLowerCase:

- ✓ These methods are used to cover all the alphabetical characters in a string to either lower case or upper case.

Example:

```
String str1 = "ABcd12#$%E";
System.out.println(str1.toLowerCase()); //abcd12#$%e
System.out.println(str1.toUpperCase()); //ABCD12#$%E
```

equals:

- ✓ This method is used to compare both the strings,
  If both are equal, returns true
  else, returns false.

Syntax:

```
String1.equals(String2)
```

Example:

```
String str1 = "firefox";
String str2 = "Firefox";
if (str1.equals(str2))
        System.out.println("Both are equal");
else
        System.out.println("Both are not equal");
```

NOTE:

- ❖ While working with string for comparision always use equals or equalsingnorecase methods but not == Operator.
- ❖ Beacuse, == operator will compare the address, where as equals method compares the actual values.

Trim:

- ✓ This method is used to remove the leading and trailing spaces in a string.
  starting ending
- ✓ This method does not remove the spaces that are present in between.

Example:

```
String str1 = "  hai hello how? ";
System.out.println(str1.length());
str1 = str1.trim();
System.out.println(str1.length());
```

Output:

```
17
14
```

Class and Object

Class:

- ✓ A class is a blue-print/template of an object.
- ✓ A class can contain variables and methods.
- ✓ For a class, generally memory is not allocated.

Object:

- ✓ An Object is an instance of a class.
- ✓ The memory is allocated, when an object is created for a class.
- ✓ By using the object reference, the values can be assigned to the variables.

Syntax for creating a class:

```
class ClassName
{
        datatype variable1;
        datatype variable2;
        datatype variable3;
                |
                |
                |
}
```

Example:

```
class Student
{
        String name;
        String fathername;
        int standard;
        int id;
        char gender;
        String schoolname;
}
```

Syntax for creating an object for a class:

```
classname objectname = new classname();
```

example:

```
Student s1 = new Student();
```

- ✓ Here,when an object is created, the memory is aloocated for the variables in the below format.
- ✓ To access the assigned variable momeory locations, we need to use objectname.variablename

Example for assigning values for an object variables:

```
s1.name = "kiran";
s1.fathername = "sai";
s1.standard = 10;
s1.id = 415;
s1.gender = 'M';
s1.schoolname = "Delhi Public School";
```

Example for printing the values of an object variables:

```
System.out.println(s1.name);
System.out.println(s1.fathername);
System.out.println(s1.standard);
System.out.println(s1.id);
System.out.println(s1.gender);
System.out.println(s1.schoolname);
```

Complete code:

```
package basicJava;
public class ClassNObject
{
	public static void main(String[] args)
	{
		Student s1 = new Student();
		s1.name = "kiran";
		s1.fathername = "sai";
		s1.standard = 10;
		s1.id = 415;
		s1.gender = 'M';
		s1.schoolname = "Delhi Public School";
		System.out.println(s1.name);
		System.out.println(s1.fathername);
		System.out.println(s1.standard);
		System.out.println(s1.id);
		System.out.println(s1.gender);
		System.out.println(s1.schoolname);
	}
}
class Student
{
	String name;
	String fathername;
	int standard;
	int id;
	char gender;
	String schoolname;
}
```

Static –Non Static Variables:

Non-Static Variables:

- ✓ All the variables which do not contain static keyword in the declaration are non-static variables.
- ✓ By default variables will be non-static.
- ✓ For non-static variables memory will be allocated at object level.
- ✓ To access non-static variables, we need to use objectname[objectname.variablename]

Static Variables:

- ✓ The variables which contain, the static keyword in the declaration are static varibles.
- ✓ For static variables, the memory is allocated at the class level.
- ✓ to access static variables, we can use classname[classname.variablename]

Example Prog For static Variables:

```
package basicJava;
public class ClassNObject
{
	public static void main(String[] args)
	{
		Student s1 = new Student();
		s1.name = "kiran";
		s1.fathername = "sai";
		s1.standard = 10;
		s1.id = 415;
		s1.gender = 'M';

		System.out.println(s1.name);
		System.out.println(s1.fathername);
		System.out.println(s1.standard);
		System.out.println(s1.id);
		System.out.println(s1.gender);
		System.out.println(Student.schoolname);

		Student s2 = new Student();
		s2.name = "srinivas";
		s2.fathername = "maruthi";
		s2.standard = 9;
		s2.id = 514;
		s2.gender = 'M';

		System.out.println(s2.name);
		System.out.println(s2.fathername);
		System.out.println(s2.standard);
		System.out.println(s2.id);
		System.out.println(s2.gender);
		System.out.println(Student.schoolname);
	}
}
class Student
{
	String name;
	String fathername;
	int standard;
	int id;
	char gender;
	static String schoolname = "Delhi Public School";
}
```

Constructor:

- ✓ constructor is something which is used for constructing an object very easily

Syntax:

[modifier] ClassName([datatype argument1, datatype argument2,.....]) {

Propertyname1= argument1;

Propertyname2= argument2; . . . . . }

Inheritance:

- ✓ It is a concept provided in java which is used for extending the parent class stub to the child class.

Note:

- ❖ one class can't extend more than one classes at a time in java.
- ❖ one class can implement one or more interfaces at a time.

Overriding:

- ✓ if at all a function is existing in the parent class and the same function with same arguments we write in child class that concept is known as overriding.

Overloading:

- ✓ If at all we have the same function more than once in a class with different arguments (may be no or type of arguments) then that concept is known as overloading.

Note:

- ❖ While calling any function we can call the required values.
- ❖ A function can return a value, if at all we want to return a value in the place of void we need to write data type of function and in the body we must write a return statement.

Collections:

The following are the drawbacks of using arrays:

- ✓ Once the arrays are defined, we can cannot change the size of an array at run time.
- ✓ To verify the existence of values in an array there is no predefined method.
- ✓ It is not possible to add or remove the memory locations of the indexes in an array.
- ✓ It is not possible to verify the duplicate values automatically in an array.
- ✓ For an array the index is always an integer with consecutive values [0 to n-1]
- ✓ Collections can be used to overcome the drawbacks of using arrays.

Collections:

- ✓ Collections refer to a kind of dynamic arrays in java.
- ✓ If we use collections, elements can be searched easily, removed or added easily.
- ✓ The following are the basic type of collections:
  - * ArrayList{Class} -- List{Interface}
  - * HashSet{Class} -- Set{Interface}
  - * HashMap{Class} -- Map{Interface}

Array List:

- ✓ Array list can have can have duplicate values.
- ✓ Import the following packages

```
import java.util.ArrayList;
import java.util.List;
```

- ✓ Code to declare arraylist and assign values

```
List<String> arr1 = new ArrayList<String>();
arr1.add("Kiran");
arr1.add("Arjun");
arr1.add("Avinash");
arr1.add("Arjun");
```

- ✓ Code to verify existance of value:

```
System.out.println(arr1.contains("Arjun"));
```

- ✓ Code to remove value from list

```
arr1.remove("Avinash");
```

- ✓ Code to print all the values of a list

```
for(String ele:arr1)
{
        System.out.println(ele);
}
```

- ✓ Code to add value at specified index

```
arr1.add(0, "xyz");
```

HashSet:

- ✓ Set is similar to list, but only difference is that Set does not accept duplicate values.
- ✓ Set will ignore if we are trying to assign any duplicate values.

```
public static void main(String[] args)
{
        String str1 = "abcdcaabbccccddddefa";
        char charr1[] = str1.toCharArray();
        Set<Character> arr1 = new HashSet<Character>();
        for(char c1:charr1)
        {
                if(arr1.add(c1))
                {
                        System.out.println(c1);
                }
        }
}
```

Note:

- ✓ In List and Set, the index value is an integer.
- ✓ In order to use the index value as String or any other datatype, we can use Maps.
- ✓ HashMap is the class of Map Interface.

HashMap:

- ✓ In HashMap the values can be stored in the form of key and value pairs.
- ✓ The following is the code for declaring the hashmap and using them.
- ✓ In Map, the key text should always be unique.
- ✓ The value can be duplicated.

micromax -- 5
apple -- 65
htc -- 30
samsung -- 20

Example:

```
Map<String, Integer> allitems = new HashMap<>();
allitems.put("micromax", 55);
allitems.put("apple", 65);
allitems.put("htc", 30);
allitems.put("samsung", 20);

System.out.println(allitems.get("htc"));

Map<String, Integer> allitems = new HashMap<>();
allitems.put("micromax", 55);
allitems.put("apple", 65);
allitems.put("htc", 30);
allitems.put("samsung", 20);
Set<String>allkeys = allitems.keySet();
for(String ek:allkeys)
{
        System.out.println(allitems.get(ek));
}
```