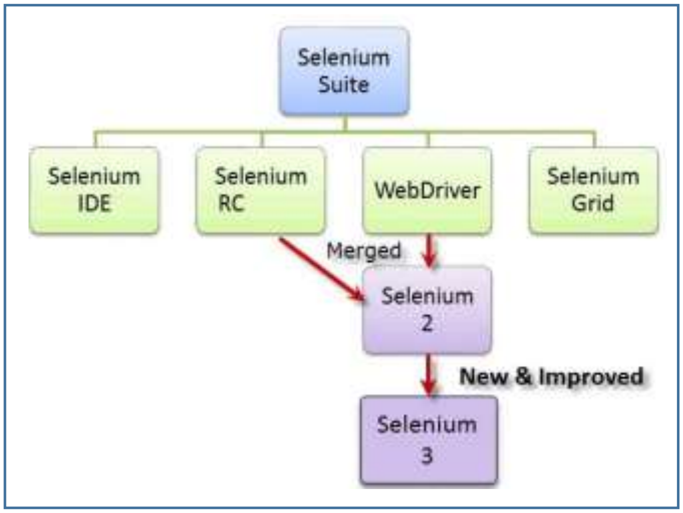
Selenium

Introduction to Selenium (Notes By – Vaibhav Mhaskar)

Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms. Selenium is not just a single tool but a suite of software's, each software is for different testing needs of an organization. It has four components.



1. **Selenium (IDE)**
2. **Selenium Remote Control (RC)**
3. **Selenium WebDriver**
4. **Selenium Grid**

Primarily, Selenium was **created by Jason Huggins in 2004**. An engineer at ThoughtWorks, he was working on a web application that required frequent testing. Having realized that the repetitious Manual Testing of their application was becoming more and more inefficient, he created a JavaScript program that would automatically control the browser's actions. He named this program as the "**JavaScriptTestRunner**."

Seeing potential in this idea to help automate other web applications, he made JavaScriptRunner open-source which was later re-named as **Selenium Core**.

So, Why the Name Selenium?

It came from a joke which Jason cracked one time to his team. Another automated testing framework was popular during Selenium's development, and it was by the company called **Mercury Interactive** (yes, the company who originally made QTP before it was acquired by HP). Since Selenium is a well-known antidote for Mercury poisoning, Jason suggested that name. His teammates took it, and so that is how we got to call this framework up to the present.

At the moment, Selenium RC and WebDriver are merged into a single framework to form **Selenium 2**. Selenium 1, by the way, refers to Selenium RC.

Selenium IDE

Selenium Integrated Development Environment (IDE) is the **simplest framework** in the Selenium suite and is **the easiest one to learn**. It is a **Firefox plugin** that you can install as easily as you can with other plugins. However, because of its simplicity, Selenium IDE should only be used as a **prototyping tool**. If you want to create more advanced test cases, you will need to use either Selenium RC or WebDriver.

Selenium Remote Control (Selenium RC)

Selenium RC was the **flagship testing framework** of the whole Selenium project for a long time. This is the first automated web testing tool that **allowed users to use a programming language they prefer**. As of version 2.25.0, RC can support the following programming languages: ***Java, C#, PHP, Python, Perl, and Ruby.***

Selenium WebDriver

The WebDriver proves itself to be **better than both Selenium IDE and Selenium RC** in many aspects. It implements a more modern and stable approach in automating the browser's actions. WebDriver, unlike Selenium RC, does not rely on JavaScript for Automation. **It controls the browser by directly communicating with it.** The supported languages are the same as those in Selenium RC : ***Java, C#, PHP, Python, Perl, Ruby.***

How to Choose the Right Selenium Tool for Your Need

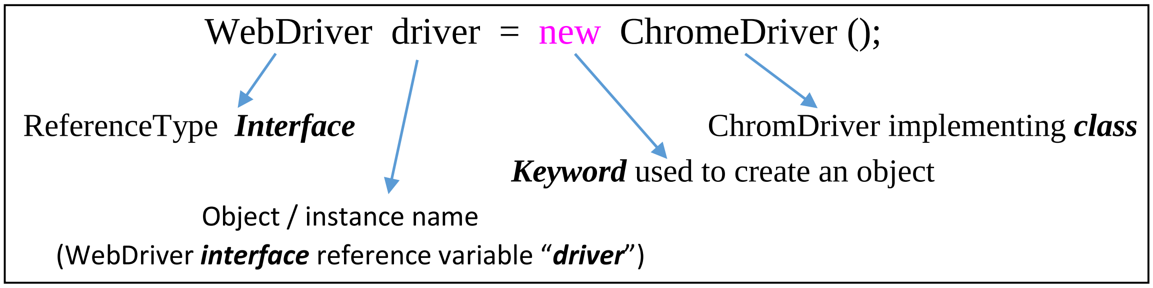
Tool	Why Choose?
Selenium IDE	• To learn about concepts on automated testing and Selenium, including. 2. Selenese commands such as type, open, clickAndWait, assert, verify, etc. • Locators such as id, name, xpath, css selector, etc. • Executing customized JavaScript code using runScript • Exporting test cases in various formats. • To create tests with little or no prior knowledge in programming. • To create simple test cases and test suites that you can export later to RC or WebDriver. • To test a web application against Firefox only.
Selenium RC	• To design a test using a more expressive language than Selenese • To run your test against different browsers (except HtmlUnit) on different operating systems. • To deploy your tests across multiple environments using Selenium Grid. • To test your application against a new browser that supports JavaScript. • To test web applications with complex AJAX-based scenarios.
WebDriver	• To use a certain programming language in designing your test case. • To test applications that are rich in AJAX-based functionalities. • To execute tests on the HtmlUnit browser. • To create customized test results.
Selenium Grid	• To run your Selenium RC scripts in multiple browsers and operating systems simultaneously. • To run a huge test suite, that needs to complete in the soonest time possible.

Summary

- The entire Selenium Tool Suite is comprised of four components:
- **Selenium IDE**, a Firefox add-on that you can only use in creating relatively simple test cases and test suites.
- **Selenium Remote Control**, also known as **Selenium 1**, which is the first Selenium tool that allowed users to use programming languages in creating complex tests.
- **WebDriver**, the newer breakthrough that allows your test scripts to communicate directly to the browser, & thereby controlling it from the OS level.
- **Selenium Grid** is also a tool that is used with Selenium RC to execute parallel tests across different browsers and operating systems.
- Selenium RC and WebDriver was merged to form **Selenium 2**.
- Selenium is more advantageous than QTP in terms of **costs and flexibility**. It also allows you to **run tests in parallel**, unlike in QTP where you are only allowed to run tests sequentially.

Selenium WebDriver is a collection of open source APIs which are used to automate the testing of a web application. ***Selenium WebDriver*** tool is used to automate web application testing scripts to verify that it works as expected. In simple words ***WebDriver*** is a tool used to automate Browser operations.

WebDriver is an interface whose methods are implemented by ***RemoteWebDriver*** class whose child classes are as follows - ChromeDriver, FirefoxDriver, IEDriver…. So on.



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

ReferenceType ***Interface***

ChromDriver implementing ***class***

Keyword used to create an object

Object / instance name

(WebDriver ***interface*** reference variable "***driver***")

WebDriver interface methods implementing classes are as follows -

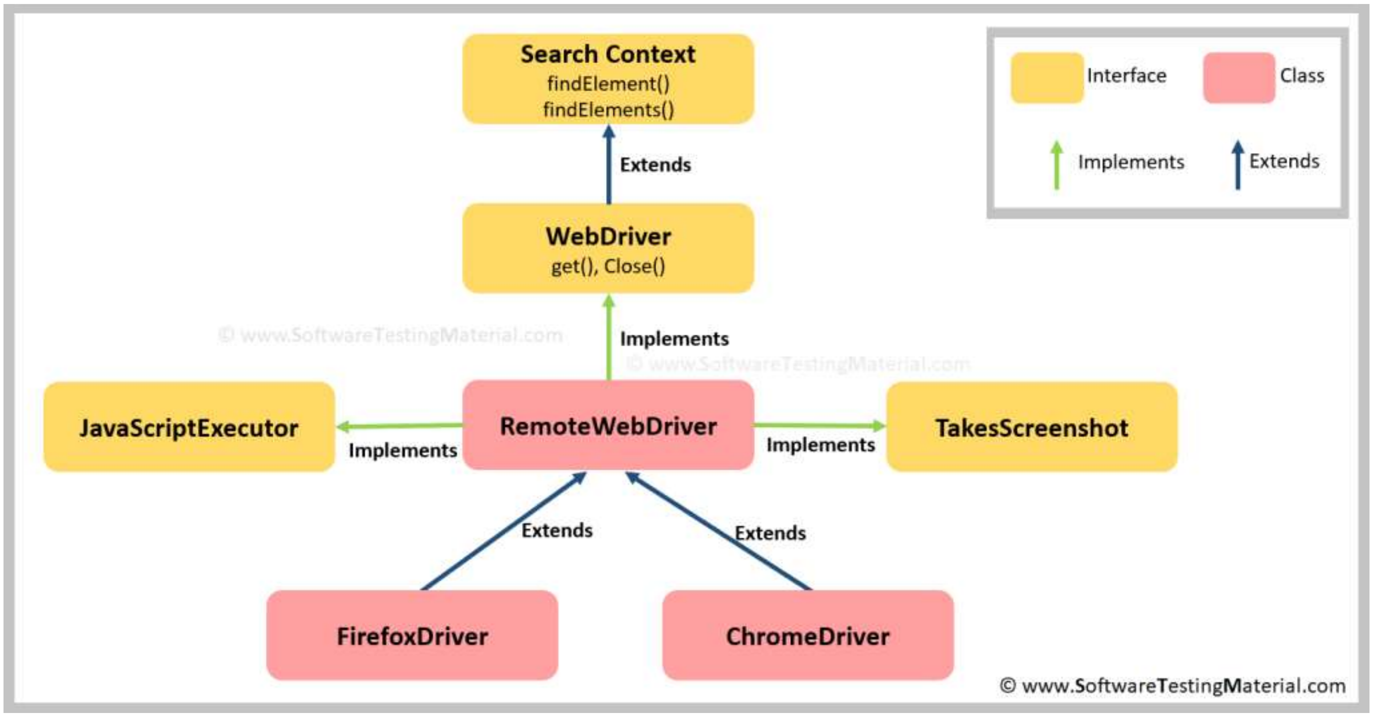
• RemoteWebDriver	• ChromeDriver	• FirefoxDriver	• InternetExplorerDriver
• OperaDriver	• SafariDriver	• EdgeDriver	• EventFiringWebDriver

The Methods of WebDriver interface are classified into following three categories –

1. Control of browser
2. Selection of WebElements
3. Debugging aids

Some Methods of WebDriver Interface are –

Sr. No.	Method	Description
1	void close()	Closes the current window
2	WebElement findElement(By by)	Find the first WebElement within the page
3	List WebElemnets findElements(By by)	Find all WebElement within the page
4	void get("http://........")	Loads the url on opened browser window
5	String getCurrentUrl()	Gets the current url of browser
6	String getPageSource()	Gets the sourse of last loaded page
7	String getTitle()	Gets the title of a page
8	String getWindowHandle()	Returns the opaque handle of this window
9	Set getWindowHandles()	Returns the set of window handles
10	Options Manage()	Get the option interface
11	Navigation Navigate()	Allows driver to access browsers History, to navigate url
12	void Quit()	Closing the browser
13	TargetLocator switchTo()	Switches to different frame or windows or alerts


Search Context
findElement()
findElements()
Extends
WebDriver
get(), Close()
Implements
JavaScriptExecutor
Implements
RemoteWebDriver
Implements
TakesScreenshot
Extends
Extends
FirefoxDriver
ChromeDriver
Interface
Class
Implements
Extends
© www.SoftwareTestingMaterial.com

Waits in Selenium (Notes By – Vaibhav Mhaskar)

- Most of the web elements are written using Java script or AJAX
- They may take certain amount of time to load on page.
- In this situation chances of getting ‘***NoSuchElementException’*** are more.
- To avoid such exceptions, Selenium has provided two types of Waits - **Implicit Wait** & **Explicit Wait.**

1. Implicit Wait:

- This wait is set for the lifetime of WebDriver instance.
- Wherever we use WebDriver instance before that line wait will be applied.
- This wait, polls for 500milliseconds (*This time depends on the browsers implementation class of webdriver. For firefox it is 500milliseconds, it depends on type and version of browser*).
- If element becomes available in first 500milliseconds then element will be returned else it will wait for next 500 milliseconds. Our script will wait for max timeout.
- We can specify default time out.
- If element is not available in timeout value then ‘***NoSuchElementException’*** will be thrown.
- We can specify timeouts in terms of: NanoSeconds, MicroSeconds, MilliSeconds, Seconds, Minutes, Hours and Day.
- E.g

```
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

- In above example timeout value is 10 Seconds.
- Now our script will wait for min 500 milliseconds and maximum 10 Second before each and every WebElement that we wish to find using ‘*driver*’ instance.
- Eg. -

```
public class ImplicitWaitDemo {
   public static void main(String[] args) {
     WebDriver driver;
     driver=new FirefoxDriver();
     driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
     driver.findElement(By.xpath("Element1")).click();
     driver.findElement(By.xpath("Element2")).click();
     driver.findElement(By.xpath("Element3")).click();
  }
}
```

In above example, our script will wait before clicking all three elements (*Element1*, *Element2 and Element3*).

- Minimum wait time is 500milliseconds and maximum wait time is 10 seconds

2. Explicit Wait:

- This wait is more customized than implicit wait.
- We can apply this wait for particular web element.
- We can even customize the polling time in explicit wait.
- We can also ignore the exceptions that we don’t want while locating an element.
- Same as implicit wait, we can set timeout for the web element.
- We can also make our script wait until certain condition occurs.

We can achieve explicit wait using two classes of Selenium: ***FluentWait*** and ***WebDriverWait.***

1. Using FluentWait:

- FluentWait class implements ***Wait*** interface.
- We can set our own ***timeout*** and ***polling time*** using FluentWait.
- We can also ***ignore exceptions*** while waiting for particular WebElement.
- We can ***wait till certain condition*** of a web element.
- This wait can be ***applied*** *to* ***particular webelement*** unlike ‘Implicit Wait’ which applied for entire life span of WebDriver instance.

Methods of FluentWait class:

1. ***withTimeout(long duration, TimeUnit unit)***

- This method is used to set maximum time that script will wait.
- This method takes two parameters: ***long duration*** and ***TimeUnit*** in terms of milliseconds, microseconds, seconds, hours, days etc.
- Eg.

```
public class FluentWaitDemo {
   public static void main(String[] args) {
      WebDriver driver;
      driver=new FirefoxDriver();
      FluentWait wait=new FluentWait(driver);
      wait.withTimeout(10, TimeUnit.SECONDS);
   }
}
```

- In above example, wait is configured for maximum 10 seconds. If element is not available in 10 seconds, then ***NoSuchElmentException*** will be thrown.

2. ***pollingEvery(long duration, TimeUnit time);***

- This method is used to set ***polling time***.
- Polling time is the frequency with which our script will check for web element.
- This method also takes two arguments: duration and time

Eg.

```
public class FluentWaitDemo {
   public static void main(String[] args) {
      WebDriver driver;
      driver=new FirefoxDriver();
      FluentWait wait=new FluentWait(driver);
      wait.pollingEvery(2, TimeUnit.SECONDS);
  }
}
```

- In above example, wait is configured for 2 seconds as polling time.
- This script will search web element after every 2 seconds till maximum timeout.

3. ***ignoring(exceptionType):***

- This method will ***ignore specific*** types of ***exceptions*** while waiting for a condition.
  Eg.

```
wait.ignoring(NoSuchElementException.class);
```

- Above script will ignore ***NoSuchElementException*** while searching a web element.

4. ***until(ExpectedConditions):***

- After configuring wait using ***until()*** method, script will keep on waiting until one of the following encounters:
  - Until Expected condition is encountered.
  - Until the timeout expires
  - Until the current thread is interrupted

Eg.

```
wait.until(ExpectedConditions.alertIsPresent());
```

- Above script will wait till alert is present on web page.

2. Using WebDriverWait:

- It is a specialized version of FluentWait.
- All its constructors take WebDriver instance as parameter.
- It inherits almost all methods of FluentWait class.
- So it is recommended to use FluentWait class when explicit wait is required.
- But when we are dealing with WebDriver instance, it is recommended to use WebDriverWait class.
- It has three constructors.

WebDriverWait(WebDriver driver, long timeOutInSeconds)

WebDriverWait (WebDriver driver, long timeOutInSeconds, long sleepInMillis)

WebDriverWait (WebDriver driver, Clock clock, Sleeper sleeper, long timeOutInSeconds, long sleepTimeOut)

Interview Questions -

1. What are the types of waits available in Selenium WebDriver?

In Selenium we could see three types of waits such as Implicit Waits, Explicit Waits and Fluent Waits.

- **Implicit Wait:** Implicit waits are used to provide a default waiting time (say 30 seconds) between each consecutive test step/command across the entire test script. Thus, subsequent test step would only execute when the 30 seconds have elapsed after executing the previous test step/command.
- **Explicit Wait:** Explicit waits are used to halt the execution till the time a particular condition is met or the maximum time has elapsed. Unlike Implicit waits, explicit waits are applied for a particular instance only.

1. What is Implicit Wait in Selenium WebDriver?

Implicit waits tell to the WebDriver to wait for a certain amount of time before it throws an exception. Once we set the time, WebDriver will wait for the element based on the time we set before it throws an exception. The default setting is 0 (zero). We need to set some wait time to make WebDriver to wait for the required time.

2. What is WebDriver Wait in Selenium WebDriver?

WebDriverWait is applied on a certain element with defined *expected condition* and *time*. This wait is only applied to the specified element. This wait can also throw an exception when an element is not found.

3. Write a code to wait for a particular element to be visible on a page. Write a code to wait for an alert to appear.

We can write a code such that we specify the XPath of the web element that needs to be visible on the page and then ask the WebDriver to wait for a specified time. Look at the sample piece of code below:

```
WebDriverWait wait=new WebDriverWait(driver, 20);

Element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath( “<xpath>”)));
```

Similarly, we can write another piece of code asking the WebDriver to wait until an error appears like this:

```
WebDriverWait wait=new WebDriverWait(driver, 20);

Element = wait.until(ExpectedConditions.alertIsPresent());
```

4. What is Fluent Wait In Selenium WebDriver?

FluentWait can define the maximum amount of time to wait for a specific condition and frequency with which to check the condition before throwing an “*ElementNotVisibleException*” exception.

5. Difference b/w implicitly Wait and Explicit wait.

No	Implicit Wait	Explicit Wait
1.	Valids for life time of webdriver instance hence once used applicable to all elements	Can apply to particular web element
2	Achieved by using **implicitlyWait**	Achieved by using **fluentWait** and **WebDriverWait**
3.	Polling time is browser & version specific and fixed and **timeouts** is **customizable**.	**Polling time** and **timeouts** is **customizable**.
4.	this wait doesn’t provide a feature to wait till a certain condition occur	this wait provides a feature to wait till a certain condition occur
5.	We **can’t ignore** any **exception** using implicit Wait	We **can ignore exception** using explicit Wait

6.	**Syntax** : driver.manage().timeouts.implicitlyWait(XX, TimeUnit.*SECONDS*);	**Syntax : 1) using FluentWait**
		FluentWait **w** = new FluentWait(driver);
		w.withTimeout(10,TimeUnit.*SECONDS*);
		Syntax : 1) using WebDriverWait
		WebDriverWait **w** = new WebDriverWait(driver, 20);
		w.until(ExpectedConditions.elementToBeClickable(By.id(“ -------- ”)));

7.	throws **NoSuchElementException** after timeout.	throws **NoSuchElementException** after timeout if not ignored

6. **What are the different types of WAIT statements in Selenium WebDriver?** ***Or the question can be framed like this:*** **How do you achieve synchronization in WebDriver?**

   There are basically two types of wait statements: **Implicit Wait** and **Explicit Wait**.

Implicit wait instructs the WebDriver to wait for some time by polling the DOM. Once you have declared implicit wait, it will be available for the entire life of the WebDriver instance. By default, the value will be 0. If you set a longer default, then the behavior will poll the DOM on a periodic basis depending on the browser/ driver implementation. Explicit wait instructs the execution to wait for some time until some condition is achieved. Some of those conditions to be attained are:

- elementToBeClickable
- elementToBeSelected
- presenceOfElementLocated

7. **What are the Expected Conditions that can be used in Explicit Wait ?**

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

1. alertIsPresent()
2. elementSelectionStateToBe()
3. elementToBeClickable()
4. elementToBeSelected()
5. frameToBeAvaliableAndSwitchToIt()
6. invisibilityOfTheElementLocated()
7. invisibilityOfElementWithText()
8. presenceOfAllElementsLocatedBy()
9. presenceOfElementLocated()
10. textToBePresentInElement()
11. textToBePresentInElementLocated()
12. textToBePresentInElementValue()
13. titleIs()
14. titleContains()
15. visibilityOf()
16. visibilityOfAllElements()
17. visibilityOfAllElementsLocatedBy()
18. visibilityOfElementLocated()

8. Can you write the syntax of Fluent Wait ?

```
FluentWait w = new FluentWait(driver);
w.withTimeout(10,TimeUnit.SECONDS);
```

9. Can you write syntax of Explicit Wait ?

```
WebDriverWait w = new WebDriverWait(driver, 20);
w.until(ExpectedConditions.elementToBeClickable(By.id(“  --------  ”)));
```

10. How to speed execution time of Test Script using waits ?

By using explicit wait we can wait for specified time period or till the condition to met. If condition met before the specified time period then script has not to wait for complete specified time period. In this way speed of execution will improve by using explicit wait.

Locators in Selenium

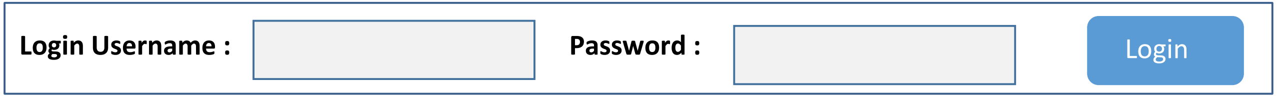
Locators are used in selenium WebDriver to find an element on a DOM. Locating elements in Selenium WebDriver is performed with the help of ***findElement()*** and ***findElements()*** methods provided by WebDriver and WebElement class.

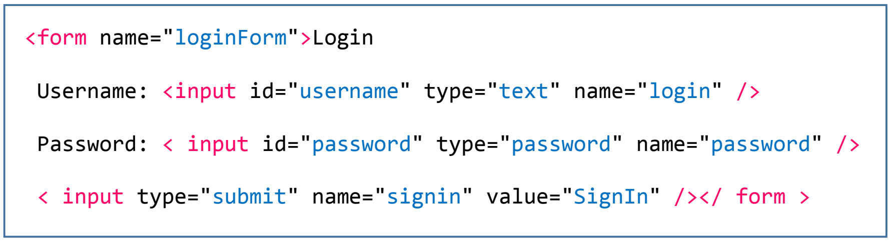
- ***findElement()*** returns a WebElement object based on a specified search criteria or ends up throwing an exception if it does not find any element matching the search criteria.
- ***findElements()*** returns a list of WebElements matching the search criteria. If no elements are found, it returns an empty list.

There are 8 types of Locators in Selenium are as follows –

Sr.	Method	Syntax	Locate By Using
1	By ID	driver.findElement(By.***id***(<element id >))	ID Attribute
2	By Name	driver.findElement(By.***name***(<element Name>))	Name Attribute
3	By LinkText	driver.findElement(By.***linkText***(<linkText >))	Link Attribute
4	By PartialLinkTest	driver.findElement(By.***partialLinkTest***(<linkText >))	Partial Link Attribute
5	By Tag Name	driver.findElement(By.***tagName***(<element HTMLTagName >))	Tag Name Attribute
6	By Class Name	driver.findElement(By.***className***(<element class>))	Class Name Attribute
7	By xPath	driver.findElement(By.***xPath***(<xpath >))	Css selector
8	By Css Selector	driver.findElement(By.***cssSelector***(<css Selector>))	xPath query

Example





```
<form name="loginForm">Login
 Username: <input id="username" type="text" name="login" />
 Password: < input id="password" type="password" name="password" />
 < input type="submit" name="signin" value="SignIn" /></ form >
```

1. Using id –

Each ***id*** is supposed to be unique couldn't be duplicated. Which makes ids a very faster and reliable way to locate elements. With ***id*** attribute value matching the location will be returned. If no element has a matching id attribute, a "**NoSuchElementException**" will be raised.

```
WebElement elementUsername = driver.findElement(By.id("username"));
WebElement elementPassword = driver.findElement(By.id("password"));
```

All objects on a page are not having id attribute, it's not realistic. In some cases developers make it having non-unique ids on a page or auto-generate the ids, in both cases it should be avoided.

2. Using Name –

By using name attribute we can find element on DOM, name attributes are not unique in a page all time. With the ***Name*** attribute value matching the location will be returned. If no element has a matching name attribute, a "**NoSuchElementException**" will be raised.

```
WebElement elementUsername = driver.findElement(By.name("username"));

WebElement elementPassword = driver.findElement(By.name("password"));
```

3. Using Link –

With this you can find elements of "**a**" tags(**Link**) with the link names. Use this when you know link text used within an anchor tag.

```
<a href="link.html">Name of the Link </a>
```

Link = "Name of the Link"

```
WebElement element = driver.findElement(By.linkText("Name of the Link"));
```

4. Using xPath –

While DOM is the recognized standard for navigation through an HTML element tree, ***XPath is the standard navigation tool for XML and an HTML document is also an XML document (xHTML).*** XPath is used everywhere where there is XML. Xpath has a fixed structure (syntax). See below –

// tag[@ attribute = 'value']

Some possible syntax are as follows –

- // tag[@attribute1 = 'value' ***and*** @attribute2 = 'value']
- // tag[@attribute1 = 'value' ***or*** @attribute2 = 'value']
- // tag[@attribute1 = 'value', ***contains***(text(),'-xxxxx-')]
- // tagP[@attribute = 'value'] // innerTagOfP[@attribute1 = 'value' and @attribute2 = 'value']

By using following ways we can select username for above example :

Xpath = //*[@id='username']

Xpath = //input[@id='username']

Xpath = //form[@name='loginForm']/input[1]

Xpath = //*[@name='loginForm']/input[1]

- **Difference Between Absolute xPath and Relative xPath –**

Sr. No.	Absolute xPath	Relative xPath
1	It uses Complete path from the Root (html) Element to the desire element.	Its not complete path from root to Element.
2	If any change is made in html code then this absolute xpath will get disturbed.	If any change is made in html code then this relative xPath will not get disturbed.
3	It is not customized xpath	It is customized type of xPath
4	It starts with / .	It starts with // .
5	It is not safe	It is safe
6	It identifies element very fast	It will take little more time in identifying the element

- **We can use Inner Text for relative xpath –**

Use ***text(),"xxxx"***, ***contains(text(),"xxxx")***, ***starts-with("xxxx")*** to customize the xpath.

// tag [text( ),"xxxx"]

// tag [contains(text( ),"xxxx")]

// tag [starts-with(@id, "msg")]

How to find xpath Dynamic Element ?

Dynamic elements are those elements who changes is attribute on every runtime. Xpath Axes are used to find the xpath of the such dynamic elements.

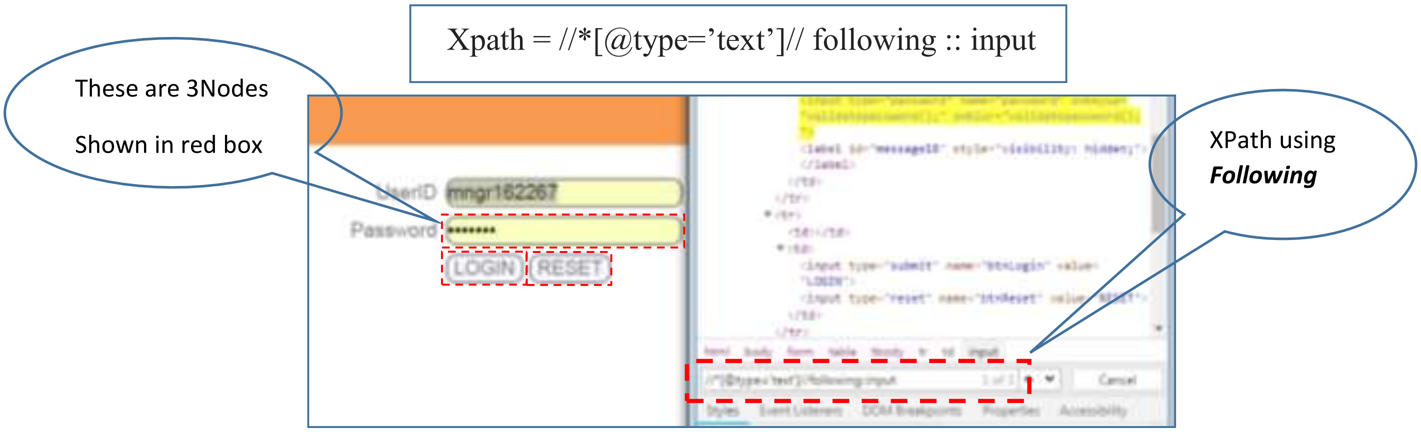
- **Xpath Axes –**

XPath Axes are the methods used to find dynamic elements. XPath axes search different nodes in XML document from current context node. XPath expression select nodes or list of nodes on the basis of attributes like ID , Name, Classname, etc. from the XML document .

a) Following:

Selects all elements in the document of the current node( ) in following image, ***input box*** is the current node.

Xpath = //*[@type='text']// following :: input



//*[@type='text']//following::input

There are 3 "input" nodes matching by using "following" axis- password, login and reset button. If you want to focus on any particular element then you can use the below XPath method:

Xpath = //*[@type='text']// following :: input[1] → Password TextBox

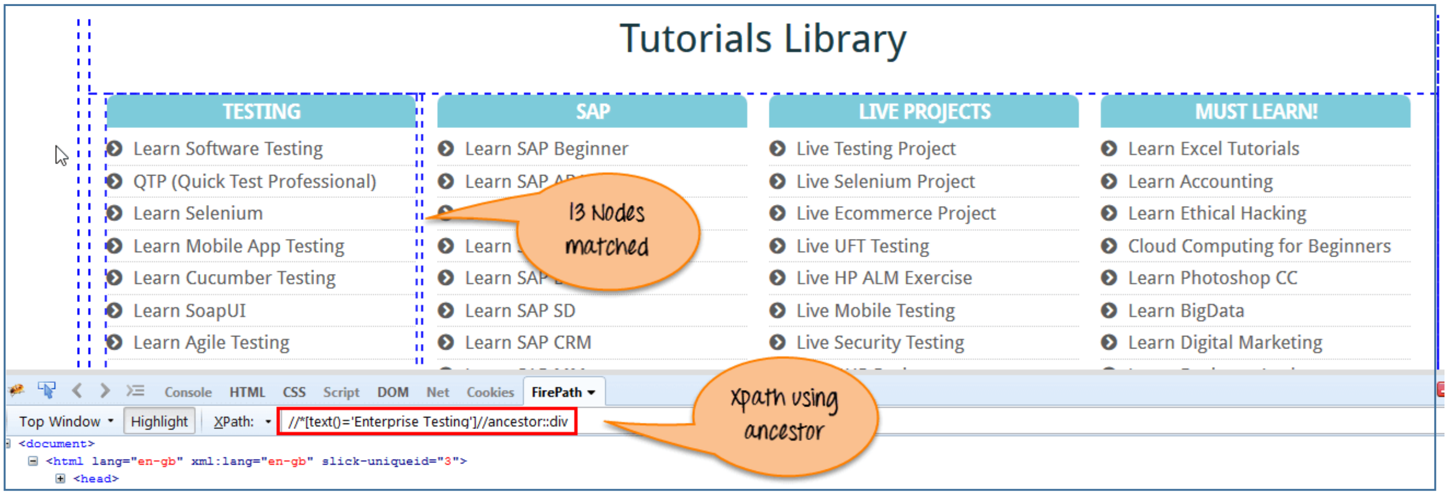
Xpath = //*[@type='text']// following :: input[2] → Login Button

Xpath = //*[@type='text']// following :: input[3] → Reset Button

b) Ancestor: पूर्वज

The ancestor axis selects all ancestors element (***parent, grandparent***,…etc.) of the current node as shown in the below screen. In the below expression, we are finding ancestors element of the current node ("ENTERPRISE TESTING" node).

Xpath = //*[text() = 'Enterprise Testing'] // ancestor :: div



TESTING
- Learn Software Testing
- QTP (Quick Test Professional)
- Learn Selenium
- Learn Mobile App Testing
- Learn Cucumber Testing
- Learn SoapUI
- Learn Agile Testing

SAP
- Learn SAP Beginner
- Learn SAP AP
- Learn
- Learn SAP
- Learn SAP SD
- Learn SAP CRM

LIVE PROJECTS
- Live Testing Project
- Live Selenium Project
- Live Ecommerce Project
- Live UFT Testing
- Live HP ALM Exercise
- Live Mobile Testing
- Live Security Testing

MUST LEARN!
- Learn Excel Tutorials
- Learn Accounting
- Learn Ethical Hacking
- Cloud Computing for Beginners
- Learn Photoshop CC
- Learn BigData
- Learn Digital Marketing

//*[text()='Enterprise Testing']//ancestor::div

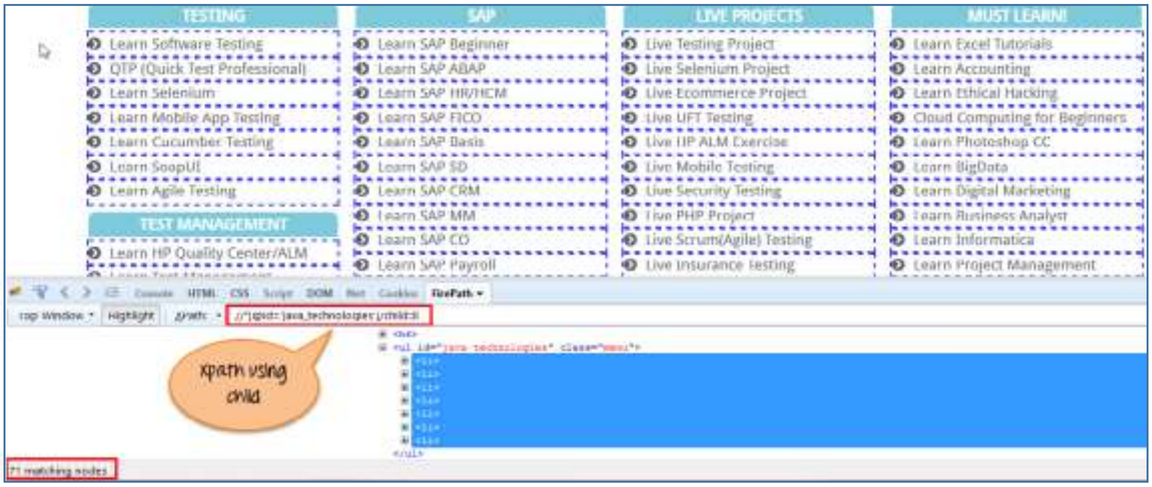
```
<document>
  <html lang="en-gb" xml:lang="en-gb" slick-uniqueid="3">
    <head>
```

There are 13 "div" nodes matching by using "ancestor" axis. If you want to focus on any particular element then you can use the below XPath, where you change the number 1, 2,3,…13 as per your requirement:

Xpath = //*[text() = ‘Enterprise Testing’] // ancestor :: div[**1**]

c) Child: Selects all children elements of the current node (Java) as shown in the below screen.

Xpath = //*[@id=’java_technologies’]/child::li



XPath: //*[@id='java_technologies']/child::li

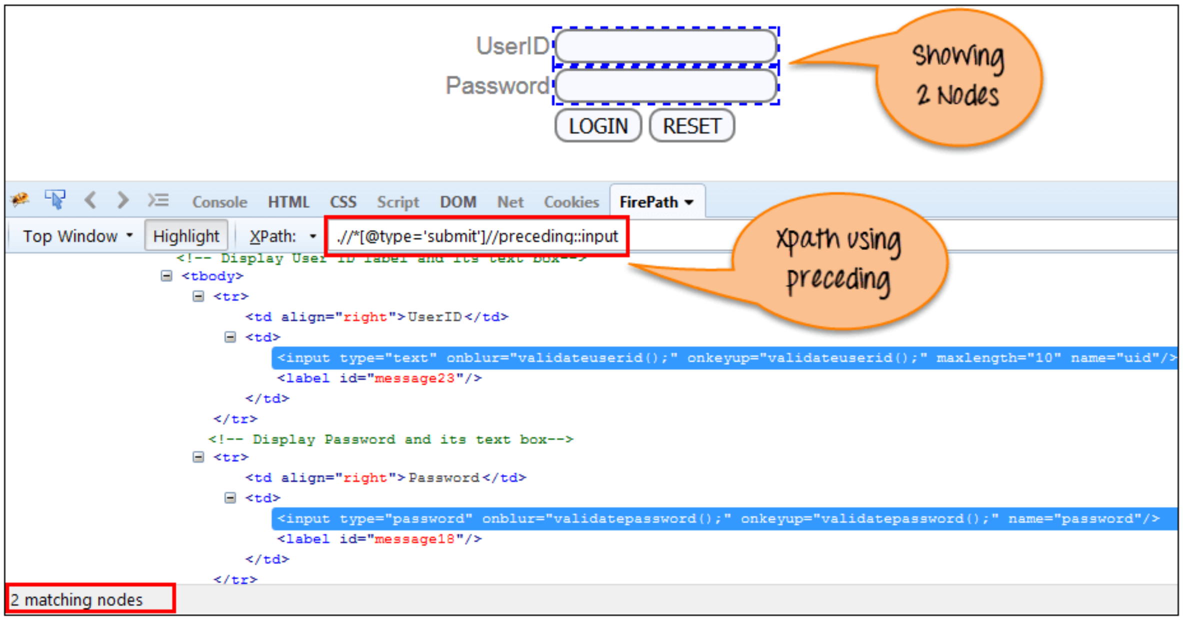
71 matching nodes

There are 71 "li" nodes matching by using "child" axis. If you want to focus on any particular element then You can change the xpath according to the requirement by putting [1],[2]…………and so on.

Xpath = //*[@id=’java_technologies’]/child::li[1]

d) Preceding: पूर्वीचे

Select all nodes that come before the current node as shown in the below screen. In the below expression, it identifies all the input elements come before "***LOGIN***" button that is **Userid** and **password** input element.


UserID
Password
LOGIN
RESET
Showing 2 Nodes
Console HTML CSS Script DOM Net Cookies FirePath
Top Window Highlight XPath:
Xpath using preceding

XPath: .//*[@type='submit']//preceding::input

```
<!-- Display User ID label and its text box-->
<tbody>
  <tr>
    <td align="right">UserID</td>
    <td>
      <input type="text" onblur="validateuserid();" onkeyup="validateuserid();" maxlength="10" name="uid"/>
      <label id="message23"/>
    </td>
  </tr>
  <!-- Display Password and its text box-->
  <tr>
    <td align="right">Password</td>
    <td>
      <input type="password" onblur="validatepassword();" onkeyup="validatepassword();" name="password"/>
      <label id="message18"/>
    </td>
  </tr>
```

2 matching nodes

```
Xpath = .//*[@type = ‘submit’]//preceding::input
```

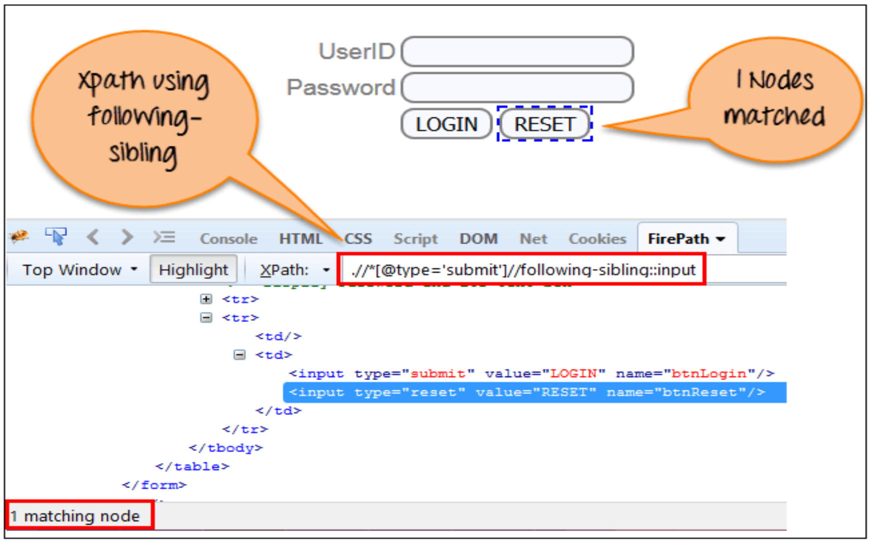
There are 2 "input" nodes matching by using "preceding" axis. If you want to focus on any particular element then You can change the xpath according to the requirement by putting [1],[2]…………and so on.

```
Xpath = //*[@type = ‘submit’]//preceding::input [1]
```

e) Following-sibling: खालील भावंड

Select the following siblings of the context node. Siblings are at the same level of the current node as shown in the below screen. It will find the element after the current ***Login*** node . One input nodes matching by using "following-sibling" axis

```
Xpath = //*[@type = ‘submit’]// following-sibling::input
```



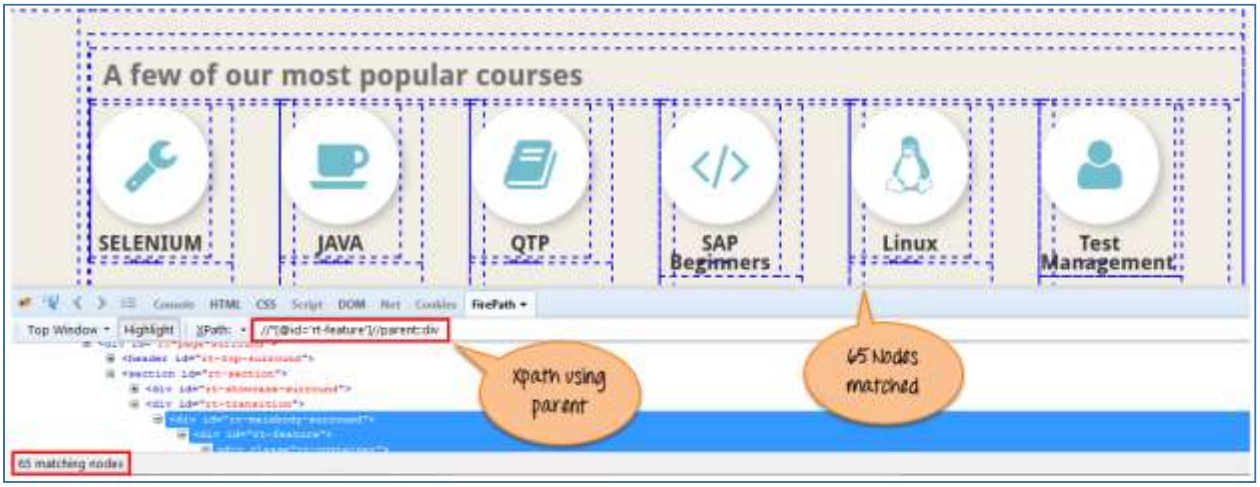
```
.//*[@type='submit']//following-sibling::input
```

```
⊞ <tr>
⊟ <tr>
    <td/>
  ⊟ <td>
      <input type="submit" value="LOGIN" name="btnLogin"/>
      <input type="reset" value="RESET" name="btnReset"/>
    </td>
  </tr>
</tbody>
</table>
</form>
```

1 matching node

f) Parent: Selects the parent of the current node as shown in the below screen.

```
Xpath = //*[@id=’rt-feature’]//parent::div
```



```
//*[@id='rt-feature']//parent::div
```

```
⊞ <div id="rt-page-surround">
  ⊞ <header id="rt-top-surround">
  ⊟ <section id="rt-section">
    ⊞ <div id="rt-showcase-surround">
    ⊟ <div id="rt-transition">
      ⊟ <div id="rt-mainbody-surround">
        ⊟ <div id="rt-feature">
          ⊞ <div class="rt-container">
```

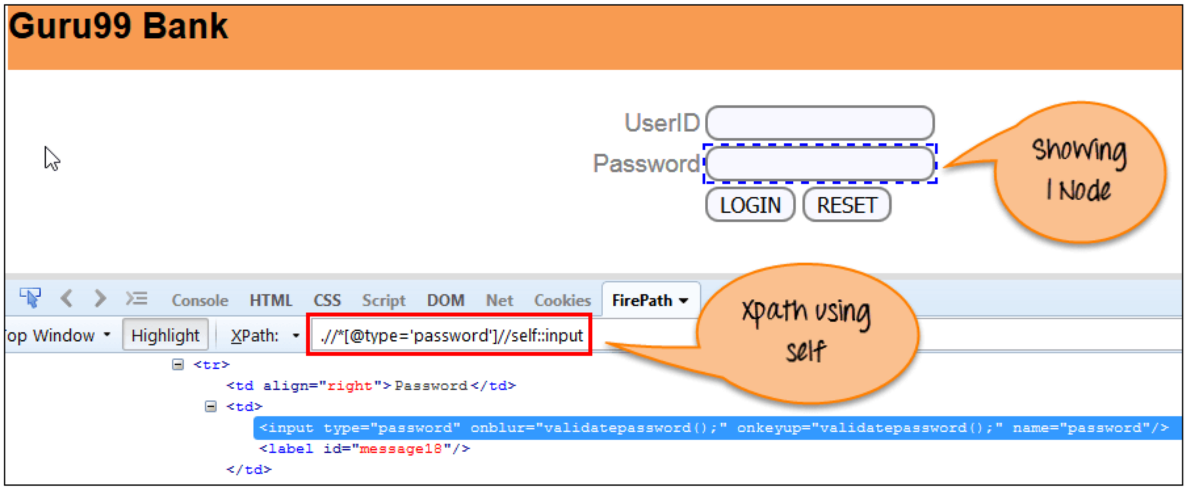
65 matching nodes

There are 65 "div" nodes matching by using "parent" axis. If you want to focus on any particular element then You can change the XPath according to the requirement by putting [1],[2]………..and so on.

Xpath = //*[@id ='rt-feature']//parent::div[1]

g) Self:

Selects the current node or 'self' means it indicates the node itself as shown in the below screen. One node matching by using "self " axis. It always finds only one node as it represents self-element.



.//*[@type='password']//self::input

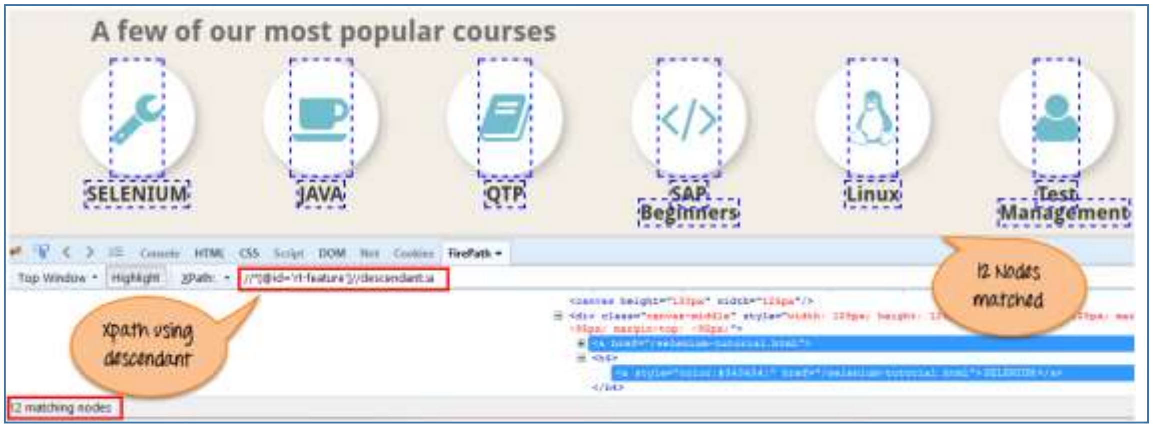
```
<tr>
    <td align="right">Password</td>
  <td>
      <input type="password" onblur="validatepassword();" onkeyup="validatepassword();" name="password"/>
      <label id="message18"/>
    </td>
```

Xpath = //*[@type = 'password']//self::input

h) Descendant:

Selects the descendants of the current node as shown in the below screen. In the below expression, it identifies all the element descendants to current element ('Main body surround' frame element) which means down under the node (child node , grandchild node, etc.).

Xpath = //*[@id = 'rt-fearture']//descendant::a



//*[@id='rt-feature']//descendant::a

```
<canvas height="132px" width="125px"/>
<div class="canvas-middle" style="width: 220px; height: 1
-50px; margin-top: -50px;">
  <a href="/selenium-tutorial.html">
  <h4>
      <a style="color:#343434;" href="/selenium-tutorial.html">SELENIUM</a>
  </h4>
```

12 matching nodes

There are 12 "link" nodes matching by using "descendant" axis. If you want to focus on any particular element then You can change the XPath according to the requirement by putting [1],[2]………..and so on.

Xpath = //*[@id = ‘rt-fearture’]//descendant::a [1]

5. Using CSS Selector -

There is a debate on the performance of CSS Locator and XPath locator. Most of the automation testers believe that using CSS selectors makes the execution of script faster compared to XPath locator. CSS Selector locator is always the best way to locate elements on the page. CSS is always same irrespective of browsers.

CSS selector structure is - *Tag [attribute = “value”]*

In dynamic elements, there is always a part of locator which is fixed. We need to generate the locator using this fixed part

If fixed part is at starting → use (^) → e.g. input [id^=’XXXXXX’]

If fixed part is at mid → use (*) → e.g. input [id*=’XXXXXX’]

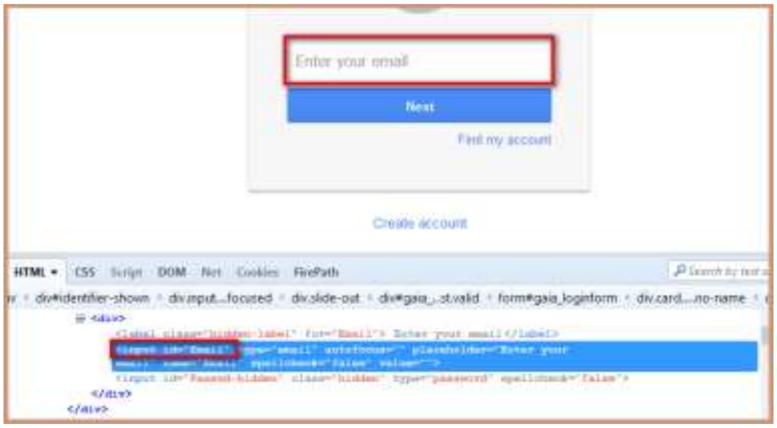
If fixed part is at end → use ($) → e.g. input [id$=’XXXXXX’]

Following are some of the mainly used formats of CSS Selectors.

- Tag and ID
- Tag and Class
- Sub-String Matches
  - Starts With (^)
  - Ends With ($)
  - Contains (*)
- Tag and Attribute
- Tag, Class, and Attribute
- Child Elements
  - Direct Child
  - Sub-child
  - nth-child

- **Tag and ID :**

Syntax: css=tag#id


Enter your email
Next
Find my account
Create account
HTML CSS Script DOM Net Cookies FirePath
Search by text
div#identifier-shown div.input...focused div.slide-out div#gaia_...st.valid form#gaia_loginform div.card....no-name

```
<div>
    <label class="hidden-label" for="Email"> Enter your email</label>
    <input id="Email" type="email" autofocus="" placeholder="Enter your
    email" name="Email" spellcheck="false" value="">
    <input id="Passwd-hidden" class="hidden" type="password" spellcheck="false">
  </div>
</div>
```

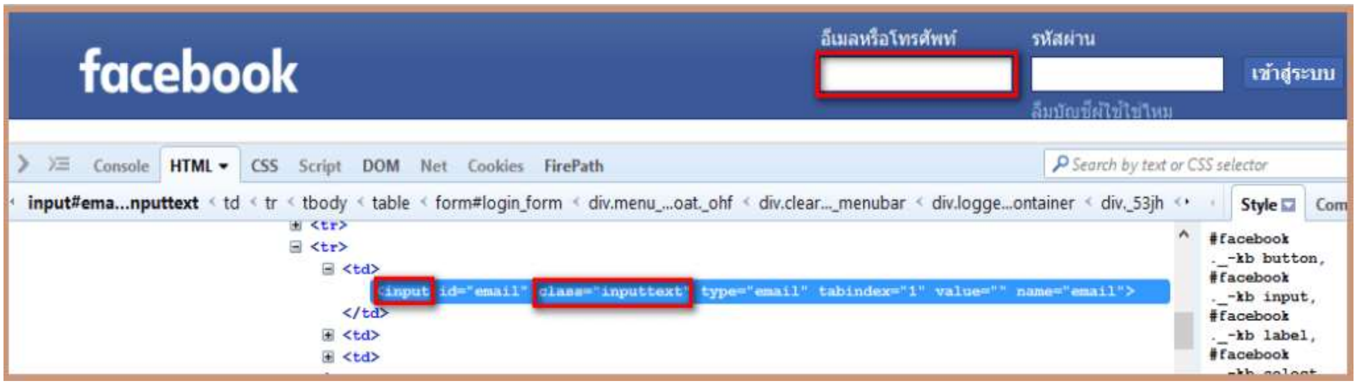
```
<div>
<label class=”hidden-label” for=”Email”> Enter your email</label>
<input id=”Email” type=”email” autofocus=”” placeholder=”Enter your email” name=”Email”
spellcheck=”false” value=””> <input id=”Passwd-hidden” class=”hidden” type=”password”   spellcheck=”false”>
</div>
```

Css = input#Email

- Tag and Class:

If multiple elements have the same HTML tag and class, then the first one will be recognized.

Syntax: css=tag.class



```
<tr>
<tr>
  <td>
    <input id="email" class="inputtext" type="email" tabindex="1" value="" name="email">
  </td>
  <td>
  <td>
```

```
#facebook
._-kb button,
#facebook
._-kb input,
#facebook
._-kb label,
#facebook
```

```
<td>

<input id=”email“ class=”inputtext“ type=”email“ tabindex=”1“ value=”” name=”email“>

</td>
```

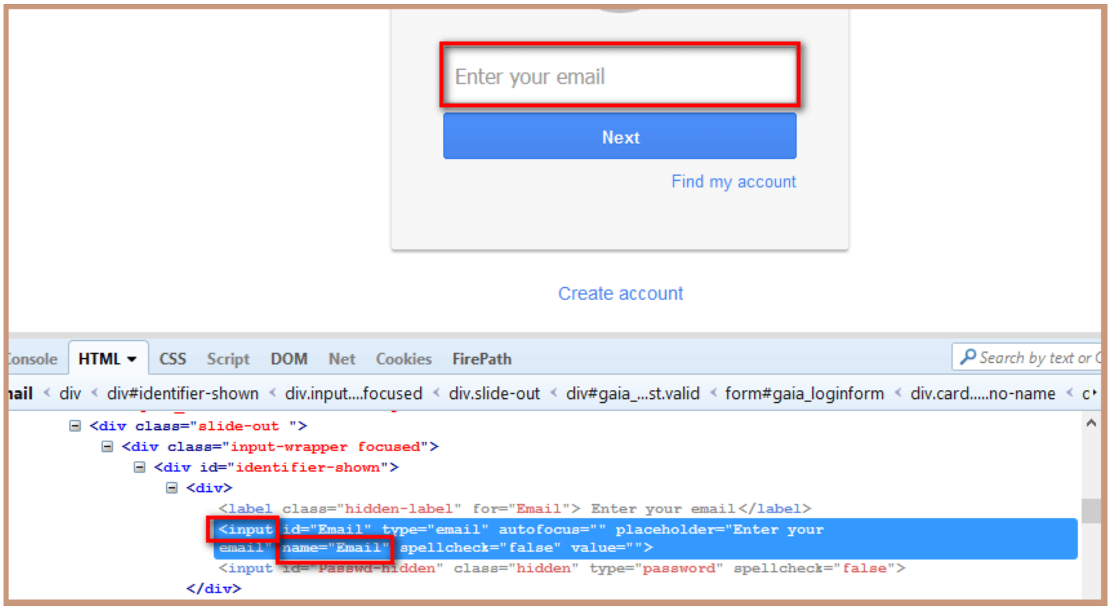
css=input.inputtext

- Tag and Attribute:

If multiple elements have the same HTML tag and attribute, then the first one will be recognized. It acts in the same way of locating elements using CSS selectors with the same tag and class.

Syntax: css=tag[attribute=value]

```
<div>

<label class=”hidden-label“ for=”Email“> Enter your email</label>

<input id=”Email“ type=”email“ autofocus=”” placeholder=”Enter your email“ name=”Email“ spellcheck=”false“ value=””> <input id=”Passwd-hidden“ class=”hidden“ type=”password“ spellcheck=”false“>

</div>
```

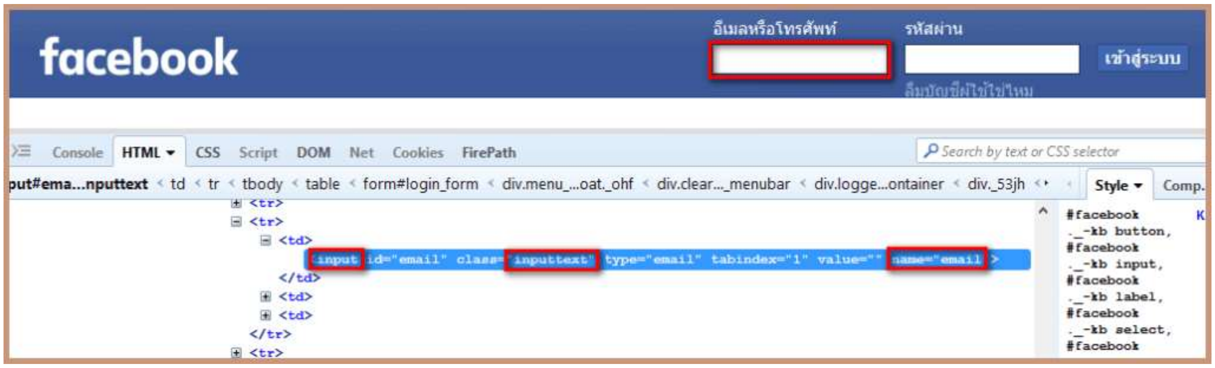

Enter your email
Next
Find my account
Create account
Console
HTML
CSS
Script
DOM
Net
Cookies
FirePath
Search by text or C
ail < div < div#identifier-shown < div.input....focused < div.slide-out < div#gaia_...st.valid < form#gaia_loginform < div.card.....no-name < c

```
<div class="slide-out ">
  <div class="input-wrapper focused">
    <div id="identifier-shown">
      <div>
        <label class="hidden-label" for="Email"> Enter your email</label>
        <input id="Email" type="email" autofocus="" placeholder="Enter your
        email" name="Email" spellcheck="false" value="">
        <input id="Passwd-hidden" class="hidden" type="password" spellcheck="false">
      </div>
```

css = input[name=Email]

Tag, Class And Attribute:

Syntax: css=tag.class[attribute=value]


facebook
อีเมลหรือโทรศัพท์
รหัสผ่าน
เข้าสู่ระบบ
ลืมบัญชีผู้ใช้ใช่ไหม
Console
HTML
CSS
Script
DOM
Net
Cookies
FirePath
Search by text or CSS selector
put#ema...nputtext < td < tr < tbody < table < form#login_form < div.menu_...oat._ohf < div.clear..._menubar < div.logge...ontainer < div._53jh
Style
Comp.

```
<tr>
<tr>
  <td>
    <input id="email" class="inputtext" type="email" tabindex="1" value="" name="email" >
  </td>
  <td>
  <td>
</tr>
<tr>
```

```
#facebook
._-kb button,
#facebook
._-kb input,
#facebook
._-kb label,
#facebook
._-kb select,
#facebook
```

```
<td>

<input id="email" class="inputtext" type="email" tabindex="1" value="" name="email">

</td>
```

css=input.inputtext[name=email]

SUB-STRING MATCHES:

CSS in Selenium has an interesting feature of allowing partial string matches using ^, $, and *.

Suppose

```
<input="Employee_ID_001">
```

Starts with (^): To select the element, we would use ^ which means ‘starts with*’*

Syntax: css=<HTML tag><[attribute^=prefix of the string]>

```
css=input[id^='Em']
```

Ends with ($): To select the element, we would use $ *which means* ‘ends with’.

Syntax: css = <HTML tag> <[attribute$=suffix of the string]>

```
css=input[id^=’001’]
```

*Contains (*): To select the element, we would use * which means* ‘sub-string*’*

Syntax: css=<HTML tag><[attribute*=sub string]>

```
css=input[id*='id']
```

Also we can use ‘contains()’:

```
Css = "input:contains('id')"
```

Locating Child Elements(Direct Child):

```
<div id="buttonDiv" class="small">
<button id="submitButton" type="button" class="btn">Submit</button>
</div>
```

Syntax: parentLocator>childLocator

CSS Locator: div#buttonDiv>button

Explanation: ‘div#buttonDiv>button’ will first go to div element with id ‘buttonDiv’ and then select its child element – ‘button’

Locating elements inside other elements (child or sub-child):

Syntax: parentLocator>locator1 locator2

CSS Locator: div#buttonDiv button

Explanation: 'div#buttonDiv button' will first go to div element with id 'buttonDiv' and then select 'button' element inside it (which may be its child or sub child)

Locating nth Child: To find nth-child css.

```
<ul id="automation">

  <li>Selenium</li>

  <li>QTP</li>

  <li>Sikuli</li>

</ul>
```

To locate the element with text 'QTP', we have to use "nth-of-type"

```
css="ul#automation li:nth-of-type(2)"
```

Similarly, To select the last child element, i.e. 'Sikuli', we can use

```
css="ul#automation li:last-child"
```

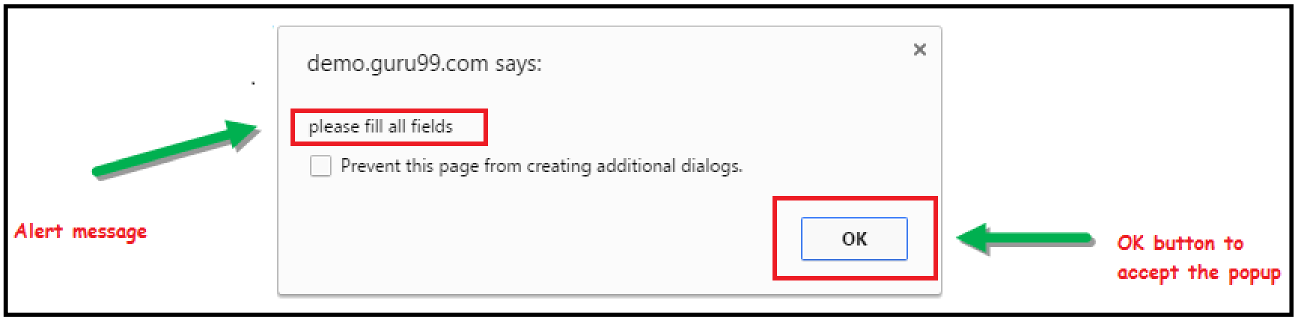
What is Alert?

Alert is a small message box which displays on-screen notification to give the user some kind of information or ask for permission to perform certain kind of operation. It may be also used for warning purpose.

Here are few alert types:

1) Simple Alert -

This simple alert displays some information or warning on the screen.



demo.guru99.com says:

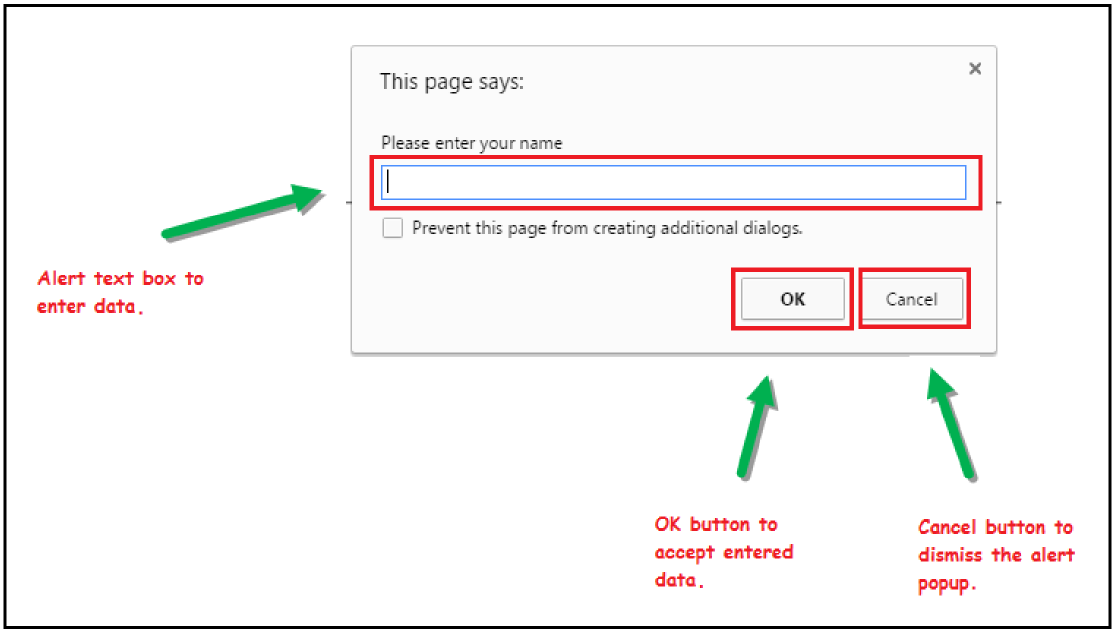
please fill all fields

□ Prevent this page from creating additional dialogs.

Alert message

OK button to accept the popup

2) Prompt Alert - This Prompt Alert asks some input from the user and selenium webdriver can enter the text using sendkeys(" input…. ").



This page says:

Please enter your name

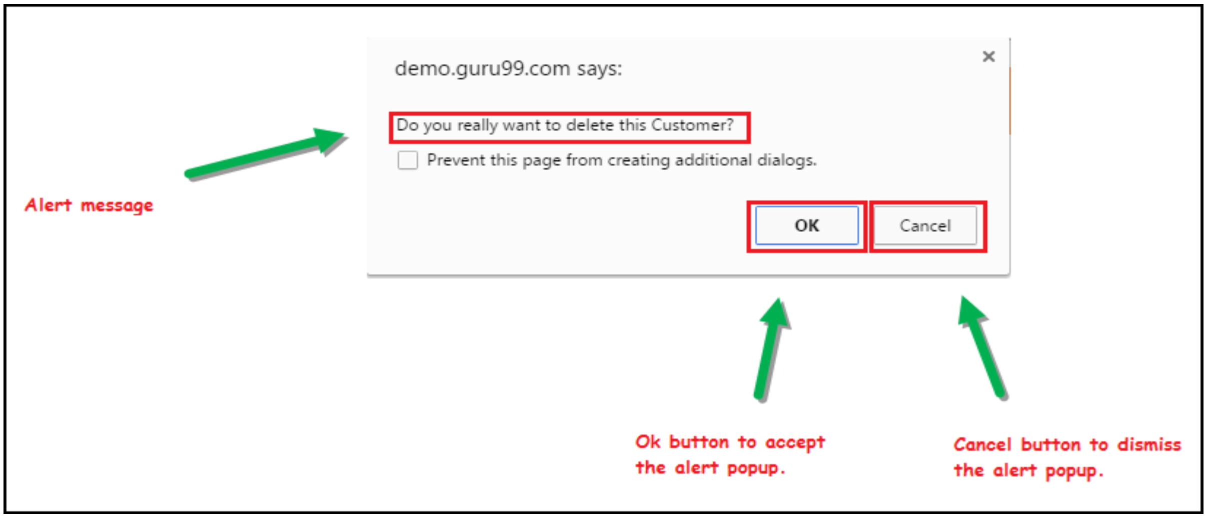
□ Prevent this page from creating additional dialogs.

Alert text box to enter data.

OK button to accept entered data.

Cancel button to dismiss the alert popup.

3) Confirmation Alert - This confirmation alert asks permission to do some type of operation.



demo.guru99.com says:

Do you really want to delete this Customer?

□ Prevent this page from creating additional dialogs.

Alert message

Ok button to accept the alert popup.

Cancel button to dismiss the alert popup.

How to handle Alert in Selenium WebDriver ?

Alert interface provides the below few methods which are widely used in Selenium Webdriver.

Sr. No.	Methods	Description
1	driver.switchTo().alert().***dismiss()***	To click on the 'Cancel' button of the alert.
2	driver.switchTo().alert().***accept()***	**To click on the 'OK' button of the alert.**
3	driver.switchTo().alert().***getText()***	**To capture the alert message.**
4	driver.switchTo().alert().***sendKeys(String “xxxx”)***	**To send some data to alert box.**

Example Code -

```
import org.openqa.selenium.NoAlertPresentException;
import org.openqa.selenium.Alert;
  public class AlertDemo {
   public static void main(String[] args) throws NoAlertPresentException,InterruptedException  {
        System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");
             WebDriver driver = new ChromeDriver();
             driver.get("http://demo.guru99.com/test/delete_customer.php");
             driver.findElement(By.name("cusid")).sendKeys("53920");
             driver.findElement(By.name("submit")).submit();
             Alert alert = driver.switchTo().alert();                    // Switching to Alert
             String alertMessage= driver.switchTo().alert().getText(); // Capturing alert message.
             System.out.println(alertMessage);                         // Displaying alert message
             Thread.sleep(5000);
             alert.accept();                                           // Accepting alert
    }
  }
```

iframes Handling (Notes By – Vaibhav Mhaskar)

iFrame is a HTML document embedded inside an HTML document. ***iFrame*** is defined by an <iframe> </iframe> tag in HTML. With this tag you can identify an ***iFrame*** while inspecting the HTML tree.

Here is an sample HTML code of a HTML page which contains two iFrames :

```
<html>
 <body>
  <div class="box">
   <iframe name="iframe1" id="IF1" height="600" width="400" src="http://toolsqa.wpengine.com"></iframe>
  </div>
<div class="box">
  <iframe name="iframe2" id="IF2" height="600" width="400" align="left" src="htp//demoqa.com"></iframe>
</div>
 </body>
</html>
```

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

1. ***switchTo.frame(int frameNumber)*****:** Pass the frame ***index*** and driver will switch to that frame.

2. ***switchTo.frame(string frameNameOrId)*****:** Pass the ***frame element Name*** or ***ID*** and driver will switch to that frame.
3. ***switchTo.frame(WebElement frameElement)*****:** Pass the frame ***web element*** and driver will switch to that frame.

How to get total number of frames on a webpage?

There are two ways to find total number of ***iFrames*** in a web page. First by executing a ***JavaScript*** and second is by finding total number of ***web elements with a tag name of iFrame***. Here is the code using both these methods:

```
WebDriver driver = new FirefoxDriver();
driver.get("http://toolsqa.wpengine.com/iframe-practice-page/");

//By executing a java script
JavascriptExecutor exe = (JavascriptExecutor) driver;
Integer numberOfFrames = Integer.parseInt(exe.executeScript("return window.length").toString());
System.out.println("Number of iframes on the page are " + numberOfFrames);

//By finding all the web elements using iframe tag
List<WebElement> iframeElements = driver.findElements(By.tagName("iframe"));
System.out.println("The total number of iframes are " + iframeElements.size());
```

Switching on iFrames

to switch to *0th iframe* by **index** - ***driver.switchTo().frame(0)***.

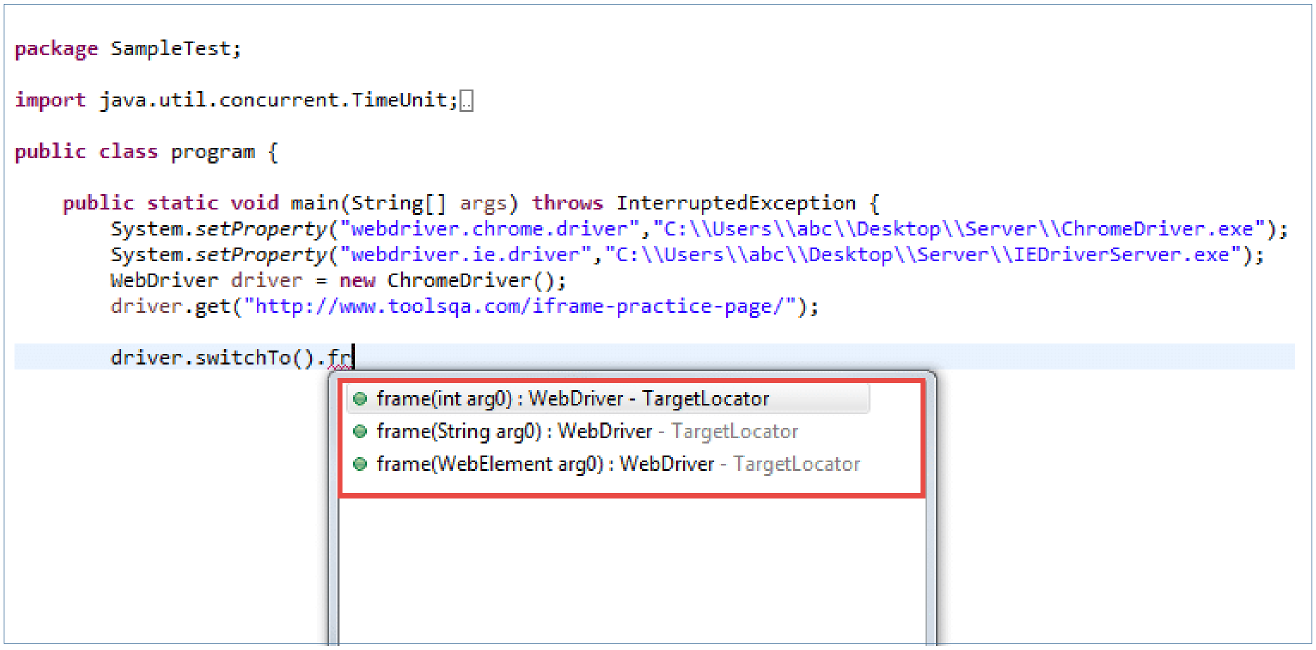
to switch *iframe* by **name** - ***driver.switchTo().frame("iframe1")***

to iframe by **WebElement** - **WebElement iframeElement = driver.findElement(By.id("IF1"));**
driver.switchTo().frame(iframeElement);

Switching back to Main Page from frame :

driver.switchTo().defaultContent();

Sample Program -



```
package SampleTest;

import java.util.concurrent.TimeUnit;

public class program {

    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver","C:\\Users\\abc\\Desktop\\Server\\ChromeDriver.exe");
        System.setProperty("webdriver.ie.driver","C:\\Users\\abc\\Desktop\\Server\\IEDriverServer.exe");
        WebDriver driver = new ChromeDriver();
        driver.get("http://www.toolsqa.com/iframe-practice-page/");

        driver.switchTo().fr
```

frame(int arg0) : WebDriver - TargetLocator
frame(String arg0) : WebDriver - TargetLocator
frame(WebElement arg0) : WebDriver - TargetLocator

Cookies Handling (Notes By – Vaibhav Mhaskar)

Practice site - (http://demo.automationtesting.in/Register.html)

Cookies are small files which are stored on a user's computer. They are designed to hold a modest amount of data specific to a particular client and website, and can be accessed either by the web server or the client computer. This allows the server to deliver a page tailored to a particular user, or the page itself can contain some script which is aware of the data in the cookie and so is able to carry information from one visit to the website (or related site) to the next.

Selenium Commands for Cookies

1. driver.manage().***getCookies()***; // returns the **set** of Cookies
2. driver.manage().***getCookieNamed(arg0)***; //returns named cookie
3. driver.manage().***addCookie(arg0)***; // create and add the cookie
4. driver.manage().***deleteCookie(arg0)***; // Delete specific cookie
5. driver.manage().***deleteCookieNamed(arg0)***; // delete named cookie
6. driver.manage().***deleteAllCookies()***; // delete all cookies

Code -

```
import java.util.Set;
import org.openqa.selenium.Cookie;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class HandlingCookies {
   public static void main(String[] args) {
          System.setProperty("webdriver.chrome.driver", "F:\\chromedriver.exe");
          WebDriver driver = new ChromeDriver();
          driver.get("https://www.amazon.in");
         Set <Cookie> cookies = driver.manage().getCookies();      // getting all cookies
          System.out.println("size of cookies : "+ cookies.size());     // printing cookies total size
         for (Cookie cookie : cookies) {
               Sopln(cookie.getName() + " : " + cookie.getValue()); // printing all cookies with value
          }
          Sopln(driver.manage().getCookieNamed("csm-hit"));       // getting "csm-hit" cookie
          Cookie cookieObj = new Cookie("myCookie","12345");  // creating obj og cookie to add new cookie
          driver.manage().addCookie(cookieObj);                          // adding cookie
         Set <Cookie> cookies1 = driver.manage().getCookies();  // after adding cookie getting all cookie
         for (Cookie cookie : cookies1) {
                Sopln(cookie.getName()+" : "+ cookie.getValue()); // printing all cookies
           }
        driver.manage().deleteCookie(cookieObj);                       // deleting cookie
        cookies1 = driver.manage().getCookies();
        Sopln("size of cookies : "+ cookies1.size());                   // printing size of a cookies after deleting cookie
        for (Cookie cookie : cookies1) {
               Sopln(cookie.getName()+" : "+cookie.getValue()); // printing all cookies
        }
       driver.quit();
}}
```

Mouse Click & Keyboard Event :

Handling special keyboard and mouse events are done using the **Advanced User Interactions API**. It contains the **Actions** class and the **Action** interface that are needed when executing these events. The following are the most commonly used keyboard and mouse events provided by the Actions class.

Sr. No	Method	Description
1	clickAndHold()	Click and hold at current mouse pointer location
2	contextClick()	Does context click (Right click) at current mouse pointer location
3	doubleClick()	Double Click at current mouse pointer location
4	dragAndDrop(source, target)	Click and hold a source element, moves to target location releases the mouse
5	dragAndDrop(source , x-offset, y-offset)	Click and hold a source element, moves to offset location & releases the mouse
6	keyDown(modifier_key)	Press the modifier key, doesn't release the modifier key press, subsequent interactions may assumed, it is kept pressed.
7	keyUp(modifier_key)	Releases key
8	moveByOffset(x-offset, y-offset)	Moves the mouse pointer to offset position
9	moveToElement(toElement)	Move the mouse pointer to the middle of element
10	release()	Release the pressed mouse left button to current location
11	sendKey(onElement, CharSequence)	Sends a character sequence to the element (text box)

Action interface and Actions Class :

Actions class implement the Action interface. Action interface have only one method: **perform()** **.** The action method gets the arguments as constructor and then implementing class decides what interaction should be done on the webpage. For examples finding an Element, passing keys using ***sendkeys*** and highlighting it. Import Action and Actions class library.

Then configure it by creating an object of Actions class like below :

```
Actions act = new Actions(driver);
act.moveToELement(toElement).click().perform();
```

Example Code 1 – Entering the text

```
public class demo {
        public static void main(String[] args) throws InterruptedException {
                System.setProperty("webdriver.chrome.driver","F:\\chromedriver.exe");
                WebDriver driver = new ChromeDriver();
                driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
                driver.manage().window().maximize();
                driver.get("https://www.facebook.com/");
                WebElement firstName = driver.findElement(By.xpath("//input[@name='firstname']"));
                Actions act = new Actions(driver) ;
                act.sendKeys(firstName, "vaibhav").build().perform();
                // Action task = act.sendKeys(firstName, "vaibhav").build();
                // task.perform();
                System.out.println("task performed");
        }
}
```

Example Code 2 –

```
public class Test {
  public static void main(String[] args) throws InterruptedException {

        System.setProperty("webdriver.chrome.driver", "F:\\Java Basic\\JAR files\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.cleartrip.com/");
        driver.manage().window().maximize();
        driver.manage().timeouts().pageLoadTimeout(10, TimeUnit.SECONDS);
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        WebElement rdoBtn = driver.findElement(By.xpath("//input[@type='radio'and@value='RoundTrip']"));
        Point p = rdoBtn.getLocation();
        Actions act = new Actions(driver);
        act.moveByOffset( p.getX(), p.getY() ).click().perform();
  }
}
```

Robot Class in Java (notes By – Vaibhav Mhaskar)

Why Robot Class?

In certain Selenium Automation Tests, there is a need to control keyboard or mouse action to interact with OS windows like ***Download pop-up***, ***Alerts***, ***Print Pop-ups***, etc. or native Operation System applications like ***Notepad, Skype, Calculator***, etc.

Selenium WebDriver cannot handle these OS pop-ups / applications. In Java version 1.3 Robot Class was introduced. Robot Class can handle OS pop-ups/applications. Robot class is present in AWT package of JDK.

Advantages -

1. Robot Class can simulate Keyboard and Mouse Event
2. Robot Class can help in upload/download of files when using selenium web driver
3. Robot Class can easily be integrated with current automation framework (keyword, data-driven or hybrid)

Robot Class internal methods and usage

Robot Class methods can be used to interact with keyboard / mouse events while doing browser automation. Alternatively **AutoIT** can be used, but its drawback is that it generates an executable file (exe) which will only work on windows, so it is not a good option to use.

Some usual methods of Robot Class during web automation -

Sr. No	Method	Description
1	**_keyPress_()**	**press down arrow key of Keyboard**
2	***mousePress***()	press the right click of your mouse
3	***mouseMove***()	move mouse pointer to the specified X & Y coordinates
4	***keyRelease***()	release down arrow key of Keyboard
5	***mouseRelease***()	release the right click of your mouse

Sample code to automate common use cases using Robot Class –

1. **Creating a Robot class Object :**

   Robot robo = new Robot();

2. To press **Down ⬇ Arrow Key** of Keyboard :

   robo.keyPress(KeyEvent.VK_DOWN);

3. To press **TAB key** of Keyboard

robo.keyPress(KeyEvent.VK_TAB);

4. To press **Enter key** of keyboard

   robo.keyPress(KeyEvent.VK_ENTER);
5. To move **Mouse Pointer** of mouse

   robo.mouseMove(630,420) ; // x & y offsets (Co-ordinates)

6. To press **left Mouse Button** of mouse

   robo.mousePress(InputEvent.BUTTON1_DOWN_MASK);
7. To release **left Mouse Button** of mouse

   robo.mouseRelease(InputEvent.BUTTON1_DOWN_MASK);

Disadvantages of Robot class –

1. Keyword / mouse event will only ***works on current instance of Window***. E.g. suppose a code is performing any robot class event, and during the code execution user has moved to some other screen then keyword / mouse event will occur on that screen.
2. Most of the methods like **mouseMove** is screen resolution dependent so there might be a chance that code working on one machine might not work on other.

Example Code –

```
 public class demo {
    public static void main(String[] args) throws InterruptedException, AWTException {
        System.setProperty("webdriver.chrome.driver","F:\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driver.manage().window().maximize();
        driver.get("https://www.facebook.com/");
        WebElement emailOrPhone = driver.findElement(By.xpath("//input[@type='email']"));
        WebElement password  = driver.findElement(By.xpath("//input[@name='pass']"));
        Actions act = new Actions(driver) ;
        act.sendKeys(emailOrPhone, "pqrxyz").build().perform();
        act.sendKeys(password, "abcde").build().perform();
        Robot robo = new Robot();
        robo.keyPress(KeyEvent.VK_ENTER); // pressing Enter Key
        robo.keyRelease(KeyEvent.VK_ENTER); // releasing  Enter Key
   }
}
```

Selenium Actions or Java AWT Robot?

Selenium uses WebDriver API and sends commands to browser driver to perform actions (through the "**JSON wire protocol**"). However, Java AWT Robot uses native system events to control ***keyboard*** and ***mouse*** operations. If your aim is, browser automation, then technically you don't need to use Robot framework because the ***Actions***() functionality provided by Selenium is more than enough. But of course, there are cases when browser or native ***OS popup*** comes up like uploading/ downloading a file. This can also be solved using Robot framework- though generally there are selenium-specific solutions/workarounds that can help avoiding using Robot. The key idea of these workarounds is "since we cannot control the popups, just don't let them open".

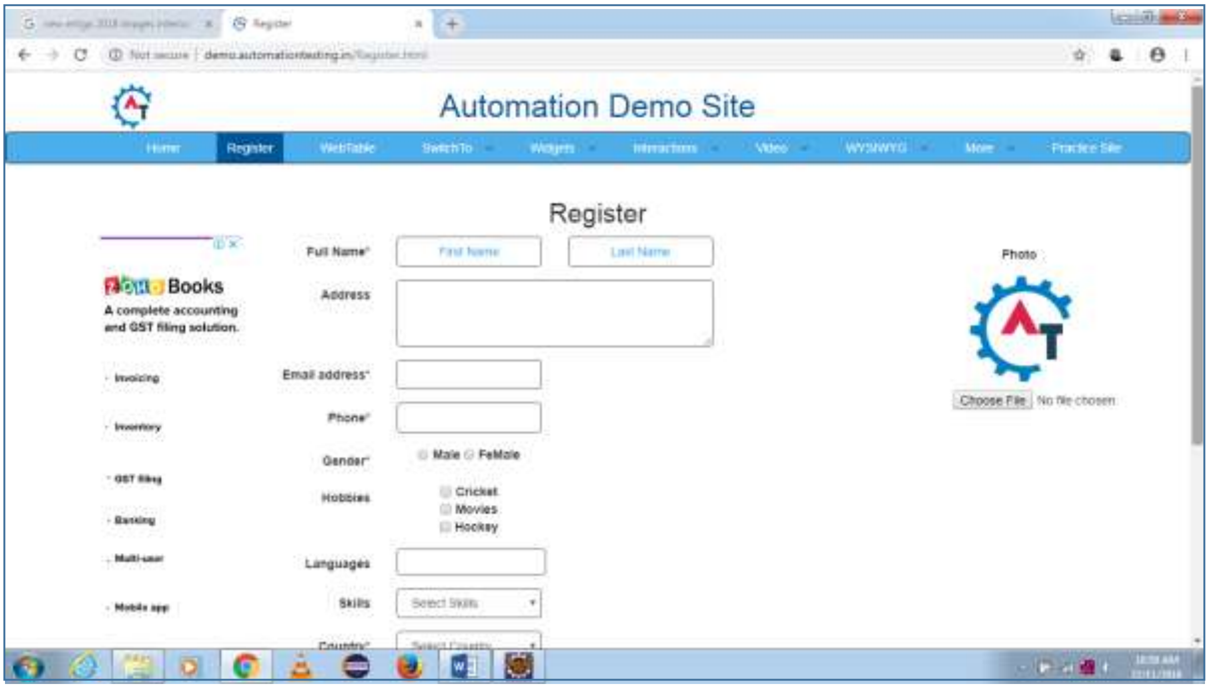
For instance, while downloading a file in Firefox, you will get a popup asking you to choose a location and filename where it should get saved. This kind a situation cannot be manipulated using selenium. But, however what you can do is, let Firefox know for certain file types, where to save the downloads automatically, without handling the popup.

Robot class is a **java based** utility which emulates the keyboard and mouse actions. The ***Actions class*** is ***selenium based*** utility, user-facing API for emulating complex user **action** events

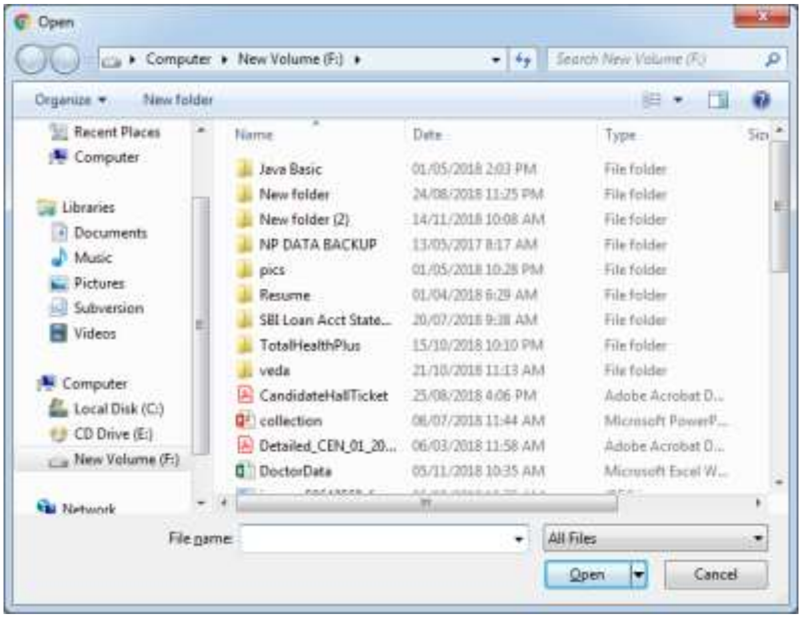
Simply we can say that **Java AWT Robot class is useful to handle OS windows like AutoIT whereas Action class is not meant to Automate OS windows**.

File uploading using AutoIT

Suppose we have to upload an image file i.e. photo after clicking on **Choose File** Button.



After Clicking on Choose Button following window based app's "**open**" window will appear which we can't automate using selenium WebDriver. Here AutoIT comes in picture. To automate Window based app we need to use AutoIT.



Name	Date	Type
Java Basic	01/05/2018 2:03 PM	File folder
New folder	24/08/2018 11:25 PM	File folder
New folder (2)	14/11/2018 10:08 AM	File folder
NP DATA BACKUP	13/05/2017 8:17 AM	File folder
pics	01/05/2018 10:28 PM	File folder
Resume	01/04/2018 6:29 AM	File folder
SBI Loan Acct State...	20/07/2018 9:38 AM	File folder
TotalHealthPlus	15/10/2018 10:10 PM	File folder
veda	21/10/2018 11:13 AM	File folder
CandidateHallTicket	25/08/2018 4:06 PM	Adobe Acrobat D...
collection	08/07/2018 11:44 AM	Microsoft PowerP...
Detailed_CEN_01_20...	06/03/2018 11:58 AM	Adobe Acrobat D...
DoctorData	05/11/2018 10:35 AM	Microsoft Excel W...

In order to automate window based app we need to download AutoIT application. i.e. AutoIT Editor and AutoIT Element Finder. Download from following links and install on your system:

AutoIT - SciTE4AutoIt3.exe : https://www.autoitscript.com/site/autoit-script-editor/downloads/

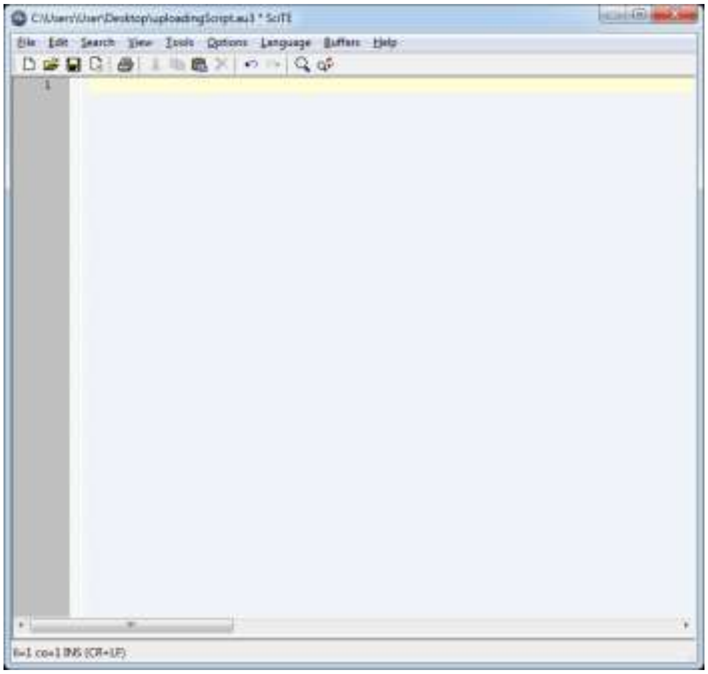
After successful installation please go to the following path on your system where can find the editor and locator

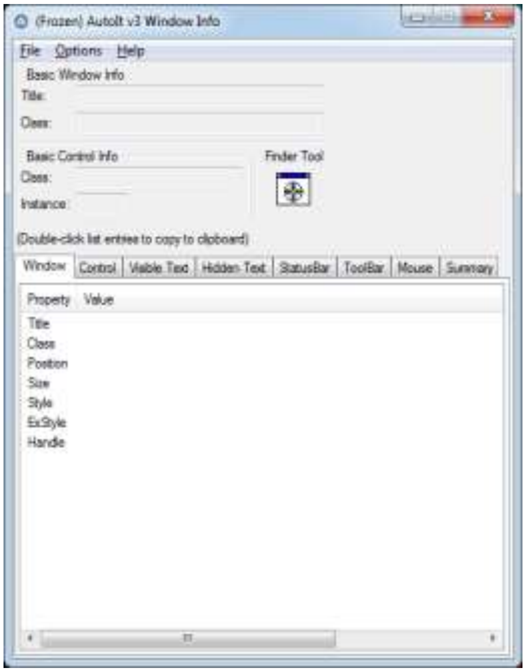
AutoIt Script Editor = "C:\Program Files (x86)\AutoIt3\SciTE-AutoIt3Installer\SciTE.exe"
AutoIt Script Editor is used to write a script to automate the window based desktop application.

AutoIt V3 Window Info = "C:\Program Files (x86)\AutoIt3\Au3Info_x64.exe"
AutoIt V3 Window Info is used to locate the element required to automate (that's why we call it Element Finder)

You may create a shortcut of above applications on your desktop screen.

This is Script Editor and This is Element Finder


C:\Users\User\Desktop\uploadingScript.au3 * SciTE
File Edit Search View Tools Options Language Buffers Help
li=1 co=1 INS (CR+LF)


(Frozen) AutoIt v3 Window Info
File Options Help
Basic Window Info
Title:
Class:
Basic Control Info
Finder Tool
Class:
Instance:
(Double-click list entries to copy to clipboard)
Window Control Visible Text Hidden Text StatusBar ToolBar Mouse Summary
Property Value
Title
Class
Position
Size
Style
ExStyle
Handle

Now let's see How to write a script in editor **:**

Following 3 command need to be executed in Editor… for desktop app automation –

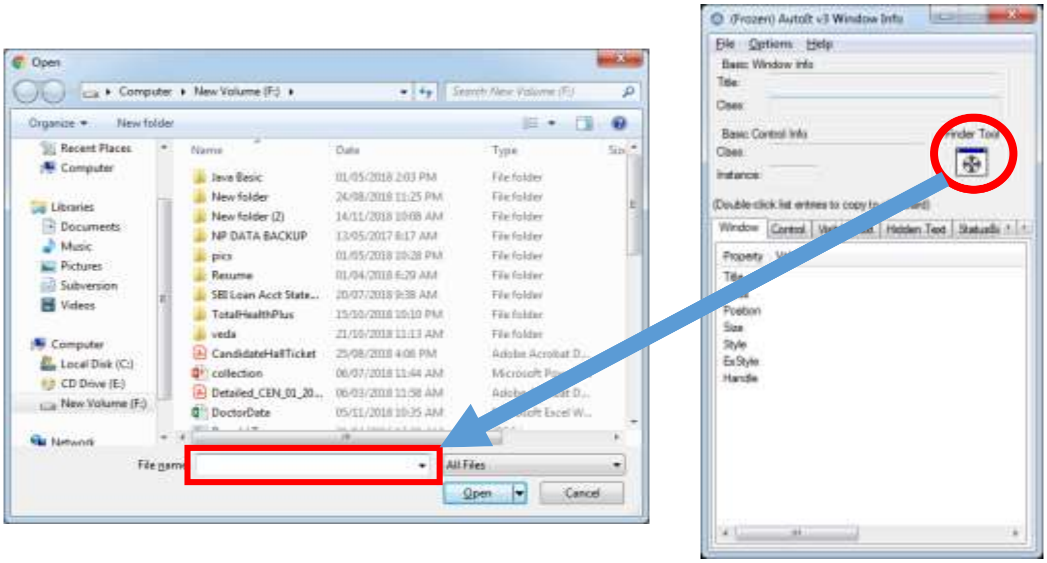
```
ControlFocus("title","text",controlID);   // Sets input focus to a given control on window

ControlSetText("title","text",controlID,"path of a file to be uploaded");  // sets test of a control

ControlClick("title","text",controlID);   //  sends a mouse click command to a given control
```

Now let's see How to search elements details for a desktop app automation script

1) Click on Choose File button to Upload the file (this action will be atomized by selenium webdriver), after clicking following widow will appear on screen whose elements details we require in desktop app automation script. Details are – title, text, controlID etc.

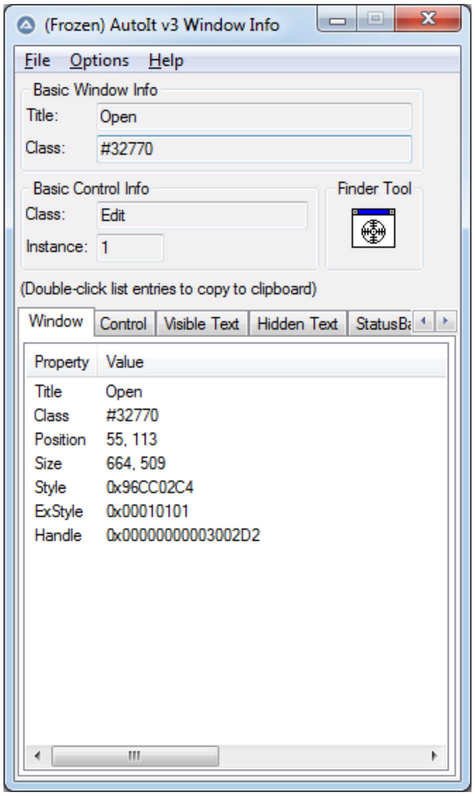
2) In following window we are going to search specific file. We just enter the path of desired file in **File Name** box.

3) In order to do so, we need to focus of automation tools attention on a **File name** box by following way – Drag and drop Finder Tool on desired element whose details we need.



Name	Date	Type
Java Basic	01/05/2018 2:03 PM	File folder
New folder	24/08/2018 11:25 PM	File folder
New folder (2)	14/11/2018 10:08 AM	File folder
NP DATA BACKUP	13/05/2017 8:17 AM	File folder
pics	01/05/2018 10:28 PM	File folder
Resume	01/04/2018 6:29 AM	File folder
SBI Loan Acct State...	20/07/2018 9:38 AM	File folder
TotalHealthPlus	15/10/2018 10:10 PM	File folder
veda	21/10/2018 11:13 AM	File folder
CandidateHallTicket	25/08/2018 4:08 PM	Adobe Acrobat D...
collection	06/07/2018 11:44 AM	Microsoft Po...
Detailed_CEN_01_20...	06/03/2018 11:58 AM	Adobe Acrobat D...
DoctorDate	05/11/2018 10:35 AM	Microsoft Excel W...

4) Then, **AutoIt v3 Window Info** (Element Finder) window will get filled with the details –


(Frozen) AutoIt v3 Window Info
File Options Help
Finder Tool
Window Control Visible Text Hidden Text StatusBa

Basic Window Info
Title: Open
Class: #32770

Basic Control Info
Class: Edit
Instance: 1

(Double-click list entries to copy to clipboard)

Property	Value
Title	Open
Class	#32770
Position	55, 113
Size	664, 509
Style	0x96CC02C4
ExStyle	0x00010101
Handle	0x00000000003002D2

Title : Open

Class : Edit

Instance : 1

Note : controlID is combination of Class + Instance , here it is **Edit1**

controlID : Edit1

Script Editor 1st command will be :

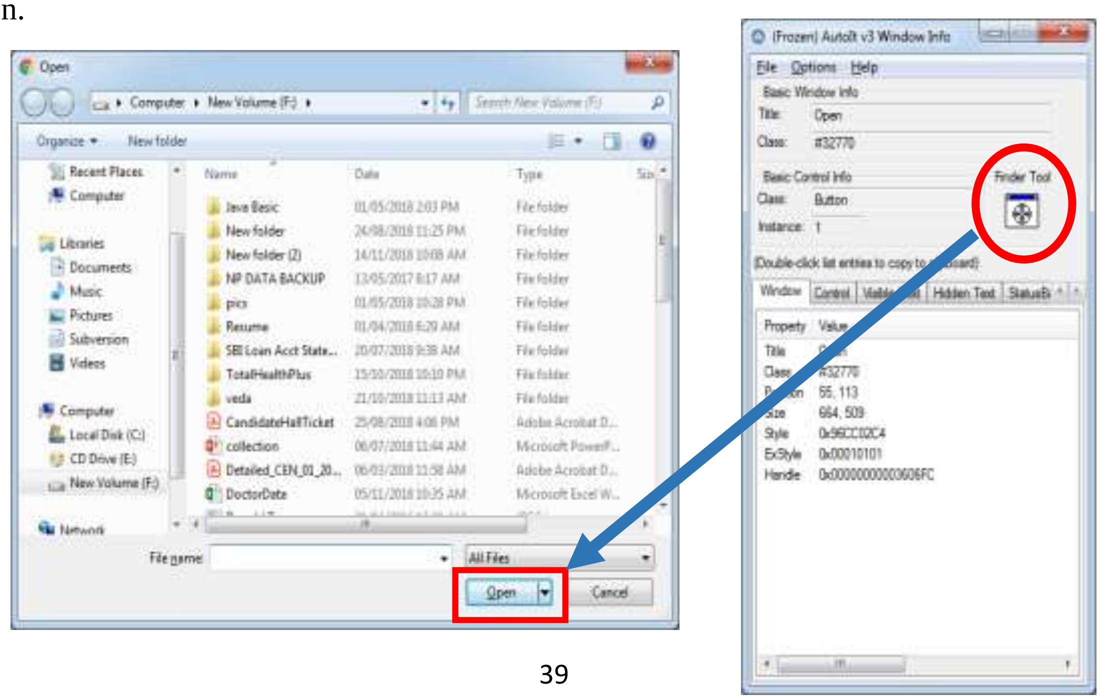
ControlFocus("Open","",EditBox1);

//here "**text**" parameter kept blank, surrounded by double quote, i.e. ""

5) And by using following command in Script Editor we are able to enter text i.e. path of file to be uploaded. 2nd command in Script Editor will be :

ControlSetText("Open","","Edit1","F:\IMG_2095 - Copy.JPG")
//here "**text**" parameter kept blank, surrounded by double quote, i.e. ""

6) Simillarly , Now do all above steps for **open Button** and find details, Drag and drop finder tool on open Button.



Name	Date	Type	Size
Java Basic	01/05/2018 2:03 PM	File folder	
New folder	24/08/2018 11:25 PM	File folder	
New folder (2)	14/11/2018 10:08 AM	File folder	
NP DATA BACKUP	13/05/2017 8:17 AM	File folder	
pics	01/05/2018 10:28 PM	File folder	
Resume	01/04/2018 6:29 AM	File folder	
SBI Loan Acct State...	20/07/2018 9:38 AM	File folder	
TotalHealthPlus	15/10/2018 10:10 PM	File folder	
veda	21/10/2018 11:13 AM	File folder	
CandidateHallTicket	25/08/2018 4:06 PM	Adobe Acrobat D...	
collection	06/07/2018 11:44 AM	Microsoft PowerP...	
Detailed_CEN_01_20...	06/03/2018 11:58 AM	Adobe Acrobat D...	
DoctorDate	05/11/2018 10:35 AM	Microsoft Excel W...	

Basic Window Info
Title: Open
Class: #32770

Basic Control Info
Class: Button
Instance: 1

(Double-click list entries to copy to clipboard)

Property	Value
Title	Open
Class	#32770
Position	55, 113
Size	664, 509
Style	0x96CC02C4
ExStyle	0x00010101
Handle	0x00000000003606FC

title : Open,

contrilID : Button1

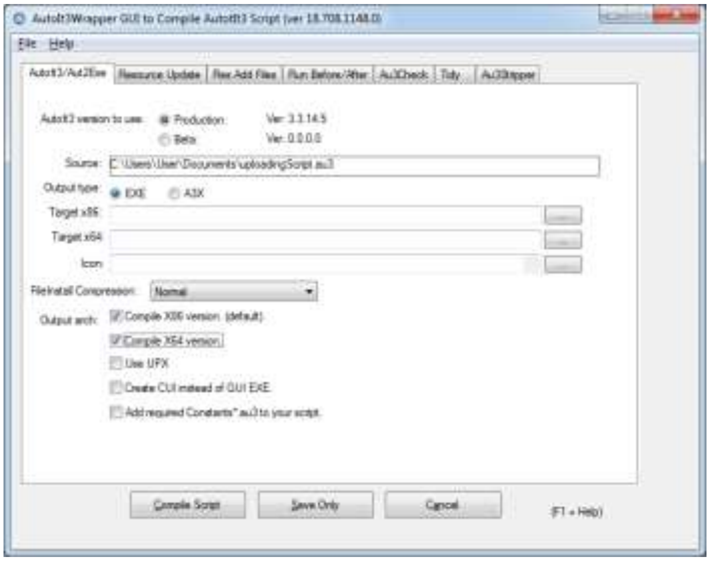
7) 3nd command in Script Editor will be :

ControlClick("Open","",Button1);   //here "**text**" parameter kept blank, surrounded by double quote, i.e. ""

8) Code in a Editor will be as follows

```
Sleep(500)
ControlFocus("Open","","Edit1")    //Sets input focus to a given control on window
Sleep(500)
ControlSetText("Open","","Edit1","F:\Donald Trump.jpg")  // sets test of a control
Sleep(500)
ControlClick("Open","","Button1")  //sends a mouse click command to a given control
Sleep(500)
```

9) and compile the code. click on **tools** and then click on **compile** then following window will appear



AutoIt3 version to use: ● Production Ver: 3.3.14.5
○ Beta Ver: 0.0.0.0

Source: C:\Users\User\Documents\uploadingScript.au3

Output type: ● EXE ○ A3X

Target x86:

Target x64:

Icon:

FileInstall Compression: Normal

Output arch: ☑ Compile X86 version (default)
☑ Compile X64 version
☐ Use UPX
☐ Create CUI instead of GUI EXE
☐ Add required Constants*.au3 to your script

- please notice the Source path
- Select **Output type** as **EXE**
- Select Output arch as Compile X64 version

(you may select Compile X64 version also depends on your systems chipset type, check for a both options) This will create a xxxxxxxx.exe file on a path mentioned in **Source.**
Check xxxxxx.exe file on **Source** path

Now we can close the Editor and Element finder and Move to Selenium Code –

In selenium script after clicking on "**Choose file**" button we have to run xxxxxxx.exe file in order to execute window app automation script by using following command –

```
Runtime.getRuntime().exec("C:\\Desktop\\uploadingScript.au3"+ " "+"F:\\Donald Trump.jpg");
```

```
//Runtime.getRuntime().exec("---xxxxx.exe file path---"+ " " +"path of file to be uploaded");
```

Selenium Script -

```
   public class AutoITDemo1 {
     public static WebElement element ;
     public static void main(String[] args) throws InterruptedException, IOException {

         String projectLocation = System.getProperty("user.dir"); // it gives project location
         System.setProperty("webdriver.chrome.driver",projectLocation+"/lib/chromedriver.exe");
         WebDriver driver = new ChromeDriver(); // launching Browser
         driver.get("http://demo.automationtesting.in/Register.html"); // loading url
         driver.manage().window().maximize();
         driver.manage().timeouts().pageLoadTimeout(40, TimeUnit.SECONDS);
         driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
         driver.findElement(By.xpath("//input[@id='imagesrc' and @onchange='uploadimg()']")).click();
         Runtime.getRuntime().exec("C:\\Desktop\\uploadingScript.au3"+ " " +"F:\\Donald Trump.jpg");
         // using this command we are executing Window app Automation Script
         //  Runtime.getRuntime().exec("---xxxxx.exe file path---"+ " " +"path of file to be uploaded");
    }
}
```

Java Script & JavaScriptExecuter (I) (Notes By – Vaibhav Mhaskar)

JavaScript is the programming language of HTML and the Web. JavaScript used to program the behavior of web page. ***JavaScriptExecutor*** is an interface of Selenium. It is used to change appearance of HTML coded webpage on a client end. Usually we design webpages using JavaScript. JavaScript written in console of a Browser. JavaScript is a programming language which is used –

- to handle elements on entire document object model (HTML document)
- Click on Element
- to enter the text in TextBox
- to select values from dropdown menu
- to handle alert
- to handle window (scrolling, Minimising, Maximizing, Closing ..)

There are two classes in JavaScript which are required to automate webpages.

1) **Document Class** –

This class refers to entire HTML DOM. We can perform different operations like ***click***, ***enter text, select from dropdown, change visibility of webElement, change colour of webElement*** etc. using this class.

Some Methods of Document class –

Sr. No	**Property/Method**	**Description**	**Syntax**
1	Cookie	Returns all name/value pairs of cookies in the document	1. document.cookie // return cookie property 2. document.cookie = *newCookie //* set new cookie
2	fullscreenEnebled()	Return a Boolean value indicating whether the document can viewed in full screen mode	Document.fullscreenEnabled
3	fullscreenElement	Return the current element that is displayed in full screen mode	var ele = fullscreenElement ;
4	getElementById()	Returns the element that has id attribute with the specified value	document.getElementById("demo");
5	getElementByClassName()	Returns a node list containing all elements with the specified class name	var x = document.getElementsByClassName("example");
6	getElementsByName()	Returns a node list containing all elements with a specified name	var x = document.getElementsByName("fname");
7	getElementByIdTageName()	Returns a node list containing all elements with the specified tag name	var x = document.getElementsByTagName("Li");
8	Title	Sets or returns the title of the document	
9	Images	Returns a collection of all <img> elements in the document	var x = document.images.length;
10	write()	Writes HTML expressions or JavaScript code to a document	document.write("Hello World");
11	writeln()	Same as write(), but adds a newline character after each statement	document.writeln("Hello World");
13	URL	Returns the full URL of the HTML Document	var x = document.URL;
14	Scripts	Returns the collection of (script) elements in the documents	var x = document.scripts.length;

2) **Window Class -**

The window object represents an open window in a browser. If a document contain frames (<**iframe**> tags), the browser creates one window object for the HTML document, and one additional window object for each frame.

Sr. No.	Method	Description	Syntax
1	alert()	Displays alert box with a message and an OK button	alert("hello, I am an Alert box !!");
2	setInterval()	Calls a function at specified intervals(milliseconds)	var myVar = setInterval(myTimer,1000);
3	setTimeout()	Calls a function after a specified number of milliseconds	

4	clearInterval()	Clears a timer set with setInterval()	var myVar = setInterval(myTimer,1000);
			clearInterval(myVar);

5	clearTimeout()	Clears a timer set with setTimeout()	mtVar=setTimeout(function(){alert("hi");},3000);
			clearTimeout(myVar);

6	close()	Closes the current window	myWIndow.close();
7	confirm()	Displays a dialog box with msg & OK & Cancel Button	confirm("Press a Button!");
8	focus()	Sets focus to current widow	myWindow.focus();
9	moveBy()	Moves a window relative to its current position	myWindow.moveBy(250,250);
10	moveTo()	Moves a window to the specified position	myWindow.moveTo(500,100);
11	open()	Opens a new browser window	Window.open("https://www.google.com");
12	print()	Prints the content of the current window	window.print();
13	resizeBy()	Resizes the window by specified number of pixels	myWindow.resizeBy(250,250);
14	resizeTo()	Resizes the window by specified width and Height	myWindow.resizeTo(250,250);
15	scrollBy()	Scrolls the document by the specified pixels	myWindow.scrollBy(100,0);
16	scrollTo()	Scrolls the document to the specified coordinates	myWindow.scrollTo(500,0);
17	stop()	Stops the widow from loading	myWindow.stop();

JavaScriptExecuter --

JavaScriptExecutor is an interface that helps us to execute Java Script through Selenium Webdriver.

Why do we need JavaScriptExecutor?

In Selenium Webdriver, locators like XPath, CSS, etc. are used to identify and perform operations on a web page. In case, these locators do not work you can use JavaScriptExecutor. You can use JavaScriptExecutor to perform a desired

operation on a web element. Selenium supports javaScriptExecutor. There is no need for an extra plugin or add-on. You just need to import **org.openqa.selenium.JavascriptExecutor** in the script as to use JavaScriptExecutor.
It has following two methods to run JavaScript on current window or on selected window.

1. **ExecutesAsynchScript(Script, Argument) –**

   With Asynchronous script, your page renders more quickly. Instead of forcing users to wait for a script to download before the page renders. This function will execute an asynchronous piece of JavaScript in the context of the currently selected frame or window in Selenium. The JavaScript so executed is single-threaded with a various callback function which runs synchronously.

2. **ExecuteScript(Script, Argument) –**

   This method executes JavaScript in the context of the currently selected frame or window in Selenium. The script used in this method runs in the body of an anonymous function (a function without a name). We can also pass complicated arguments to it.

The script can return values. Data types returned are

- Boolean
- Long
- String
- List
- WebElement

NOTE : To use JavaScript Executer we need to type caste driver instance into JavaScript executer

```
FirefoxDriver driver = new FirefoxDriver();
JavascriptExecuter JS = (JavaScriptExecuter)driver;
 JS.executeScripts(Script, arguments);
```

Example 1 – Login on Facebook using JavaScript

```
public class JavaScriptExample {
    public static void main(String[] args) {
            System.setProperty("webdriver.chrome.driver","F:\\chromedriver.exe");
            WebDriver driver = new ChromeDriver();
            driver.get("https://www.facbook.com");
            JavascriptExecutor js = (JavascriptExecutor) driver ;
            js.executeScript("document.getElementById('email').value='Vaibhav'");
            js.executeScript("document.getElementById('pass').value='Vaibhav'");
            js.executeScript("document.getElementById(\"u_0_2\").click()","");
  }}
```

Example 2 – Scrolling down using JavaScript (Interview question)

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

             System.setProperty("webdriver.chrome.driver","F:\\chromedriver.exe");
             WebDriver driver = new ChromeDriver();
             driver.get("https://www.facbook.com");
             JavascriptExecutor js = (JavascriptExecutor) driver ;
             js.executeScript("window.scrollBy(0,100)");   // scrolling down by 100
             Thread.sleep(1000);
             js.executeScript("window.scrollBy(0,100)");   // scrolling down by 100
             Thread.sleep(1000);
             js.executeScript("window.scrollBy(0,-200)");  // scrolling up by -200
 }}
```

1. TestNG Introduction
2. TestNG Annotations And Benefits
3. To Create TestNG.xml file
4. Ignore Tests in TestNG
5. Skip Test In TestNG
6. Groups
7. Exception
8. Dependencies
9. Parameterized Tests Using XML
10. Parameterized Tests Using DataProviders
11. Parallel Execution in TestNG
12. Asserts
13. Soft Assert
14. Listeners
15. IRetryAnalyzer – How To Run Failed Test Cases
16. TestNG Reports
17. How To Run TestNG Using Command Prompt
18. TestNG Reporter logs

1. TestNG Introduction

TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing to integration testing. Writing a test is typically a three-step process:

A) Write the business logic of your test and insert TestNG annotations in your code.
B) Add the information about your test (e.g. the class name, the groups you wish to run, etc) in a testng.xml file
C) Run TestNG.xml.

2. TestNG Annotations and Benefits –

Sr.No.	Annotations	Description
1	**@Test**	Marks a class or a method as a part of the test
2	**@BeforeMethod**	A method which is marked with this annotation will be executed before every ***@test*** annotated method.
3	**@AfterMethod**	A method which is marked with this annotation will be executed after every ***@test*** annotated method.
4	**@BeforeClass**	A method which is marked with this annotation will be executed before ***first @Test*** method execution. It runs only once per class.
5	**@AfterClass**	A method which is marked with this annotation will be executed after all the ***@Test*** methods in the ***current class*** have been run
6	**@BeforeTest**	A method which is marked with this annotation will be executed before ***first @Test*** annotated method.
7	**@AfterTest**	A method which is marked with this annotation will be executed when **all *@Test*** annotated methods complete the execution of ***those classes which are inside <test> tag in testng.xml file.***
8	**@BeforeSuite**	A method which is marked with this annotation will run **only once before** all tests in the suite have to run
9	**@AfterSuite**	A method which is marked with this annotation will run **once after** execution of all tests in the suite have run
10	**@BeforeGroups**	This annotated method will run **before first @test run of that specific group**.
11	**@AfterGroups**	This annotated method will run **after all @test methods** of that group completes its execution.
12	**@Parameters**	This annotation is used to pass parameters to test methods.
13	**@Listeners**	This annotation is used with test class. It helps in writing logs and results.
14	**@ Factory**	Marks a method as a factory that returns ***objects[ ]*** that will be used by TestNG as Test classes. The method must return Object[ ].
15	**@DataProvider**	we use @DataProvider annotation for data supplier method which return ***Object[][]***. which we can use in @Test annotated method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation.

***Test* – ** is a group of Test Cases (we may call it class) & ***Test Cases* – ** is a group of Test Steps i.e. methods

Note –

Class level Annotations are ***@Test***, ***@ignore***, ***@Listeners*** rest of the all annotations are method level, if we write @Test on the top of class then all the methods inside that class will be treated as test cases & they will appear in emailable-report.html file as test cases, if we write @ignore on top of class then all the methods of that class will be excluded from execution.

3. To Create TestNG.xml file

In TestNG framework, we need to create T**estng xml** file to create and handle multiple test classes. We do configure our test run, set test dependency, include or exclude any test, method, class or package and set priority etc in the xml file.

Steps to create TestNG XML file

Step 1: Create testng xml file

- Right click on Project folder, go to **New** and select '**File**' as shown in below image

- In New file wizard, add file name as '**testng xml**' as shown in below given image and click on **Finish** button.

- It will add **testng xml** file under your project folder.

Step 2 : Write xml code:

- testng.xml file format

```
 <suite>
      <test>
            <Classes>
              <class name=".test"/>
              <class name="…......>
              < .....
            </Classes>
    </test>
</Suite>
```

- Now add required code in your testng xml file.

Note: You can choose any name for your Test Suite & Test Name as per your need. Format of class name is -
class name = PackageName.ClassName

Step 3 : Execute a testng xml

- Run the test by right click on the ***testng*.*xml*** file and select ***Run As*** > ***TestNG Suite***. After execution you may check the output on console.

There is one more way to create testng.xml file in your project. It is most easy and simple way. After complete creation of your project just do following sequence

right click in your ***project*** → go to ***testNG*** → & click on ***convertTo TestNG***

and configured TestNG.xml file will be created.

Example **TestNG.xml** file of ***TotalHealthPlus*** project is as follows

```
<suite name="Suite">
            <listeners>
               <listener class name="com.thc.cases.manageNursePageTest"/ >
       </listeners>
  <test thread-count="5" name="Test">
    <classes>
      <class name="com.thc.cases.manageNursePageTest"/>
      <class name="com.thc.cases.LoginPageTest"/>
      <class name="com.thc.cases.PatientHomePageTest"/>
      <class name="com.thc.cases.DoctorAddListPageTest"/>
      <class name="com.thc.cases.AdminHomePageTest"/>
      <class name="com.thc.cases.DoctorHomePageTest"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

4. How to Ignore TestNG Test

We may include or exclude any test or method from tetsng.xml file In 3 ways,

1) By typing annotations ***(enabled= false)*** in a java code over any test or method
2) By ***excluding*** it in xml file

Code -

```
package softwareTestingMaterial;
import org.testng.annotations.Test;

public class TestCase2 {

   @Test (enabled=false)
   public void printClass2(){
     System.out.println("This is Test Case 2");
   }
}
```

TestNG.xml -

```
<suite>
   <test>
      <Classes>
         <class name=".test"/>
             <methods>
               <include name = "--- method name ---">
               <include name = "--- method name ---">
               <exclude name = "--- method name ---">
            <methods>
         <class name="……...>
         < .....
      </Classes>
   </test>
</Suite>
```

3) By typing annotation ***@ignore*** in java code over any test or method

```
package softwareTestingMaterial;
import org.testng.annotations.Test;
@ignore
public class TestCase2 {
        @Test
        public void printClass2(){
        System.out.println("This is Test Case 2");
 }   }
```

Note - There are three annotations which can be written on class level they are ***@test, @ignore, @Listeners*** rest of the all annotations are method level, if we write @Test on the top of class then all the methods inside that class will be treated as test cases & they will appear in emailable-report.html file as test cases, if we write @ignore on top of class then all the methods of that class will be excluded from execution

5. How to Skip TestNG Test

One way of skipping a test method is by using ***throw new SkipException()*** exception. Once ***SkipException***() thrown, remaining part of that test method will not be executed and control will goes directly to next test method execution. Let’s see Example code

```
public class SkipTestCase {
        @Test
       public void aSkipTest(){
           String a ="Skip Test";
           if(a.equals("Skip Test")){
               throw new SkipException("Skipping - This is not ready for testing ");
           }else{
               System.out.println("I am in else condition");
           }
          System.out.println("I am out of the if else condition");
        }
        @Test
       public void nonSkipTest(){
          System.out.println("No need to skip this test");
  }      }
```

Output on Console –

```
No need to skip this test
PASSED: nonSkipTest

SKIPPED: aSkipTest

===============================

Default suite

Total tests run: 2, Failures: 0, Skips: 1

===============================
```

6. TestNG Groups

TestNG allows you to perform groupings of test methods. We can also specify groups that contain other groups. Groups are specified in *testng.xml* file and can be found either under the <*test*> or <*suite*> tag. Groups specified in the <suite> tag apply to all the <test> tags between it.

Script – Test Case 1:

```
package softwareTestingMaterial;
import org.testng.annotations.Test;

public class TestCase1 {

  @Test(groups={"smokeTest","functionalTest"})
  public void loginTest() {
    Sopln("Logged in successfully");
  }
}
```

Script – Test Case 2:

```
package softwareTestingMaterial;
import org.testng.annotations.Test;

public class TestCase2 {

    @Test(groups={ "functionalTest" })
    public void composeMail() {
        System.out.println("Mail Sent");
    }
}
```

testng.xml :

```
<suite name="softwaretestingmaterial">
  <test name="testngTest">
           <groups>
             <define name=" smokeTest ">
                <include name="–Test Name--"/>
                <include name="–Test Name--"/>
            </define>
             <run>
               <include name="smokeTest" />
             </run>
           </groups>
    <classes>
      <class name="softwareTestingMaterial.TestCase1" />
      <class name="softwareTestingMaterial.TestCase2" />
    </classes>
  </test>
</suite>
```

Console Output :

```
[TestNG] Running:

Logged in successfully

===========================
softwaretestingmaterial
Total tests run: 1, Failures: 0, Skips: 0
===========================
```

Groups can also include other groups. These groups are called ***MetaGroups***. For example, you might want to define a group ***all*** that includes ***smokeTest*** and ***functionalTest***. Let’s modify our testng.xml file as follows:

Group of Groups:

testng.xml :

```
<suite name="softwaretestingmaterial">
  <test name="testngTest">
    <groups>
         <define name="all">
           <include name="smokeTest"/>
           <include name="functionalTest"/>
         </define>
        <run>
          <include name="all" />
        </run>
    </groups>
  <classes>
    <class name="softwareTestingMaterial.TestCase1"/>
    <class name="softwareTestingMaterial.TestCase2"/>
  </classes>
 </test>
</suite>
```

Console Output:

```
[TestNG] Running:
C:\TestNGProject\testng.xml

Logged in successfully
Mail Sent
============================
softwaretestingmaterial
Total tests run: 2, Failures: 0, Skips: 0
============================
```

We can ignore a group by using the ***<exclude>*** tag

TestNG.xml : which excluding group

```
<suite name="softwaretestingmaterial">
  <test name="testngTest">
           <groups>
                    <run>
                    <exclude name="smokeTest"/>
                    <include name="functionalTest"/>
                    </run>
          </groups>
       <classes>
          <class name="softwareTestingMaterial.TestCase1" />
          <class name="softwareTestingMaterial.TestCase2" />
      </classes>
  </test>
</suite>
```

7. TestNG Exception

Firstly, we see a basic program using TestNG Exception.

TestNG Basic Program :

```
package TestingMaterial;
import org.testng.annotations.Test;
     public class TestNGException {
          @Test
          public void testException() {
          Sopln("SoftwareTestingMaterial.com");
             int i = 1 / 0;
          }
}
```

testng.xml File :

```
 <suite name="softwaretestingmaterial">
    <test name="testngTest">
       <classes>
         <class name="TestingMaterial.TestNGException"/>
       </classes>
    </test>
</suite>
```

Console Output :

```
[TestNG] Running:

SoftwareTestingMaterial.com

===============================
Default suite
Total tests run: 1, Failures: 1, Skips: 0
===============================
```

"testException( )" method was marked as failed by TestNG during execution.

The expected exception to validate while running the below test is mentioned using the ***expectedExceptions*** attribute value while using the @Test annotations. See following example

Example Code :

```
package TestingMaterial;
import org.testng.annotations.Test;
public class TestNGException {
@Test(expectedExceptions =
ArithmeticException.class)
          public void testException() {
          Sopln("SoftwareTestingMaterial.com");
            int i = 1 / 0;
}         }
```

Testng.xml:

```
<suite name="softwaretestingmaterial">
  <test name="testngTest">
    <classes>
    <class name="TestingMaterial.TestNGException"/>
    </classes>
  </test>
</suite>
```

Console Output:

```
[TestNG] Running:


SoftwareTestingMaterial.com
=============================
Default suite
Total tests run: 1, Failures: 0, Skips: 0
=============================
```

"testException()" method was marked as passed by TestNG during execution.

8. TestNG Dependencies:

TestNG allows you to specify dependencies either with annotations or in XML. TestNG allows you to specify dependencies either with:

- Using attribute ***dependsOnMethods*** in @Test annotations, OR.
- Using attribute ***dependsOnGroups*** in @Test annotations.

Example Code :

```
import org.testng.annotations.Test;
 public class DependsOnMethodsTest {
   @Test
   public void testCase1() {
      System.out.println("Test Case 1");
   }
  @Test
   public void testCase2() {
      System.out.println("Test Case 2");
 }
}
```

Testng.xml File

```
<suite name="softwaretestingmaterial">
  <test name="testngTest">
   <classes>
   <classname="TestingMaterial.DependsOnMethodsTest"/>
   </classes>
 </test>
</suite>
```

Console Output:

```

[TestNG] Running:

Test Case 1
Test Case 2

===========================
softwaretestingmaterial
Total tests run: 2, Failures: 0, Skips: 0
============================
```

Now we add the '***dependsOnMethods***' attribute to the @Test Annotations and execute the same program.

Example Code :

```
import org.testng.annotations.Test;
 public class DependsOnMethodsTest {

@Test(dependsOnMethods={"testCase2"})
    public void testCase1() {
       System.out.println("Test Case 1");
    }
   @Test
    public void testCase2() {
       System.out.println("Test Case 2");
  }
}
```

Testng.xml File

```
<suite name="softwaretestingmaterial">
  <test name="testngTest">
   <classes>
   <className="TestingMaterial.DependsOnMethodsTest"/>
   </classes>
 </test>
</suite>
```

Execute the same testng.xml which was placed above and see the difference in Console Output

Console Output:

```
[TestNG] Running:

Test Case 2
Test Case 1
 ============================
softwaretestingmaterial
Total tests run: 2, Failures: 0, Skips: 0
============================
```

Test Case 2 executed first and the ***Test Case 1***

Let's see Dependencies with XML:

Examle Code :

```
package TestingMaterial;
import org.testng.annotations.Test;
        public class DependsOnMethodsTestCase {
                @Test(groups = { "FirstGroup" })
                public void testCase1() {
                    Sopln("Test Case 1");
                }
                 @Test(groups = { "SecondGroup" })
                 public void testCase2() {
                    System.out.println("Test Case 2");
        }       }
```

Testng.xml :

```
<suite name="Testingmaterial">

    <test name="testngTest">
       <groups>
         <dependencies>
           <group name="FirstGroup" dependson="SecondGroup"></group>
         </dependencies>
       </groups>

    <classes>
    <class name="softwareTestingMaterial.DependsOnMethodsTestCase" />
    </classes>

    </test>
   </suite>
```

```
Console Output:

[TestNG] Running:

Test Case 2
Test Case 1

============================
softwaretestingmaterial
Total tests run: 2, Failures: 0, Skips: 0
============================
```

From above console output it is clear that method assigned to dependsOn is executed first then dependent Method

9. Parameterized Tests Using XML

Parameterized tests allow developers to run the same test again and again using different values. There are two ways to set these parameters:

- with ***testng****.xml*
- with ***Data Providers***

Let’s see passing parameters with testng.xml:

With this technique, we could define the parameters in the ***testng.xml*** file and then reference those parameters in the source files.

Create a java test class, say, ***ParameterizedTest.java***

Add test method *parameterizedTest()* to your test class. This method takes a string as input parameter

Add the annotation @Parameters("browser") to this method. The parameter would be passed a value from testng.xml, which we will see in the next step.

Code

```
public class ParameterTest {
   @Test
   @Parameters("browser")
    public void parameterizedTest(String browser){
        if(browser.equals("firefox")){
          System.out.println("Open Firefox Driver");
        }else if(browser.equals("chrome")){
          System.out.println("Open Chrome Driver");
        }
 } }
```

TestNG.xml

```
<suite name="softwaretestingmaterial">
<parameter name=”browser”  value=”Firefox”/>
<test name="testngTest">
   <classes>
   <className="TestingMaterial.ParameterTest"/>
   </classes>
 </test>
</suite>
```

Here, name attribute represents the parameter name and value represents the value of that parameter. We could use this parameter to the test method ***parameterizedTest*** by using ***@Parameters("browser")*** as mentioned in the above step.

```
[TestNG] Running:

Open Firefox Driver

===============================
softwaretestingmaterial
Total tests run: 1, Failures: 0, Skips: 0
===============================
```

Note : TestNG will automatically try to convert the value specified in testng.xml to the type of your parameter. Here are the types supported:

- String
- int/Integer
- boolean/Boolean
- byte/Byte
- char/Character
- double/Double
- float/Float
- long/Long
- short/Short

10. Parameterized Tests Using DataProviders –

Specifying parameters in testng.xml might not be sufficient if you need to pass complex parameters, or parameters that need to be created from Java (complex objects, objects read from a property file or a database, etc…). In this case, you can use a Data Provider to supply the values you need to test. A Data Provider is a method in your class that returns an array of objects. This method is annotated with ***@DataProvider:***

```
public class DataProviderClass {
      // This method takes data as input parameters. The attribute dataProvider is mapped to "getData"
      @Test (dataProvider="getData") // Number of columns should match the number of input parameters
      public void loginTest(String Uid, String Pwd){
         Soln("UserName is "+ Uid);
         Soln("Password is "+ Pwd);
      }
      //If the name is not supplied, the data provider’s name automatically defaults to the method’s name.
      @DataProvider(name="getData")  //A data provider returns an array of objects.
      public Object[][] getData(){
           Object [][] data = new Object [2][2]; //Object [][] data = new Object [rowCount][colCount];

           data [0][0] = "FirstUid";
           data [0][1] = "FirstPWD";
           data[1][0] = "SecondUid";
           data[1][1] = "SecondPWD";
      return data;
      }}
```

Console Output:

```
[TestNG] Running:



 UserName is FirstUid
 Password is FirstPWD
 UserName is SecondUid
 Password is SecondPWD

 ===============================
 softwaretestingmaterial
 Total tests run: 2, Failures: 0, Skips: 0
 ===============================
```

11. Parallel Execution in TestNG –

There are situations where we want to run multiple tests with same or different browsers at the same time. In such cases, we can use “parallel” attribute in testng.xml to accomplish parallel test execution in TestNG.

The parallel attribute of suite tag can accept four values:

tests – All the test cases inside <test> tag of testng.xml file will run parallel

classes – All the test cases inside a java class will run parallel

methods – All the methods with *@Test* annotation will execute parallel

instances – Test cases in same instance will execute parallel but two methods of two different instances will run in different thread.

In the below program, I took two methods. First methods opens Firefox driver and navigate to ***http://www.SoftwareTestingMaterial.com*** and closes the browser. Second methods opens Chrome driver and navigate to the same URL and closes the browser.

```
 public class ParallelTests {
      @Test
       public void getFirefox(){
         System.setProperty("webdriver.gecko.driver", "D://geckodriver.exe");
         Sopln("GetFirefox  is running on Thread : " + Thread.currentThread().getId());
         WebDriver driver = new FirefoxDriver();
         driver.get("http://www.SoftwareTestingMaterial.com");
         driver.close();
     }

     @Test
     public void getChorme(){
       System.setProperty("webdriver.chrome.driver", "D://chromedriver.exe");
       Sopln("GetChrome is running on Thread : " + Thread.currentThread().getId());
       WebDriver driver = new ChromeDriver();
       driver.get("http://www.SoftwareTestingMaterial.com");
       driver.close();
   }
}
```

TestNG.xml file -

```
<suite name="softwaretestingmaterial" parallel="methods" thread-count="2">
            <test name="testngTest">
  <classes>
    <class name="softwareTestingMaterial.ParallelTests" />
  </classes>
            </test>

</suite>
```

Once you run the testng.xml using the above code, you could see both the browsers in action at a time.

Here in the above testng.xml file, I have passed ***parallel*=*methods*** and ***thread-count*=*2*** at the suite level. I would like to execute selenium scripts in parallel in different threads. Most of the times, these two methods will execute in different threads. Thread Id may vary on every run. Here we are just passing thread count but we are not assigning any thread id, assigning thread id will be taken care by your system processor.

12. TestNG Asserts –

TestNG Asserts help us to verify the condition of the test in the middle of the test run. Based on the TestNG Assertions, we will consider a successful test only if it is completed the test run without throwing any exception.

Assert.assertEquals(driver.getTitle(), actualTitle);

Example Code

```
public class TestNGAsserts {
            @Test
            public void testNGAsserts() throws Exception{
                        System.setProperty("webdriver.gecko.driver","D://geckodriver.exe");
            WebDriver driver = new FirefoxDriver();
 //Test Condition 1: If Page title matches with actualTitle then it finds email title and enters the value which we
pass
                      driver.get("https://www.gmail.com");
                      String actualTitle = "Gmail";
                      Assert.assertEquals(driver.getTitle(), actualTitle);
                      Thread.sleep(2000);
                      driver.findElement(By.xpath("//*[@id='Email']")).sendKeys("SoftwareTestingMaterial.com");
                      //Test Condition 2: If page title didnt match with actualTitle then script throws an exception
                      Thread.sleep(2000);
                      driver.get("https://www.gmail.com");
                      actualTitle = "GoogleMail";
                     Thread.sleep(2000);
                    //Assert.assertEquals(driver.getTitle(), actualTitle, "Title not matched");
                     Assert.assertEquals(driver.getTitle(), actualTitle);
}  }
```

Different TestNG Asserts Statements:

- ***Assert.assertEqual(String actual, String expected) :***Asserts that two Strings are equal. If they are not, an AssertionError is thrown.

  Parameters:
  actual – the actual value
  expected – the expected value

- ***Assert.assertEqual(String actual, String expected, String message)* :** Asserts that two Strings are equal. If they are not, an AssertionError, with the given message, is thrown.

  Parameters:
  actual – the actual value
  expected – the expected value
  message – the assertion error message

- ***Assert.*assertEquals(boolean actual, boolean expected) :** Asserts that two Booleans are equal. If they are not, an AssertionError is thrown.

  Parameters:
  actual – the actual value
  expected – the expected value

- ***Assert.assertTrue(condition)* :** Asserts that a condition is true. If it isn't, an AssertionError is thrown.

  Parameters:
  condition – the condition to evaluate

- ***Assert.assertTrue(condition, message) :*** *Asserts that a condition is true. If it isn't, an AssertionError, with the given message, is thrown.*

  Parameters:
  condition – the condition to evaluate
  message – the assertion error message

- ***Assert.assertFalse(condition) :*** Asserts that a condition is false. If it isn't, an AssertionError is thrown.

  Parameters:
  condition – the condition to evaluate

- ***Assert.assertFalse(condition, message) :*** Asserts that a condition is false. If it isn't, an AssertionError, with the given message, is thrown.

Parameters:
condition – the condition to evaluate
message – the assertion error message

13. Soft Assert –

There are two types of Assert **Hard Assert** & **Soft Assert**

Hard Assert –

Hard Assert throws an ***AssertException*** immediately when an assert statement fails and test suite continues with next ***@Test***. The disadvantage of Hard Assert – It marks method as fail if assert condition gets failed and the remaining statements inside the method will be aborted.

Soft Assert –

Soft Assert collects errors during *@Test*. Soft Assert does not throw an exception when an assert fails and would continue with the next step after the assert statement. If there is any exception and you want to throw it then you need to use *assertAll()* method as a last statement in the @Test and test suite again continue with next *@Test* as it is.

We need to create an object to use Soft Assert which is not needed in Hard Assert. Here we took two methods namely ***softAssert()*** and ***hardAssert().***

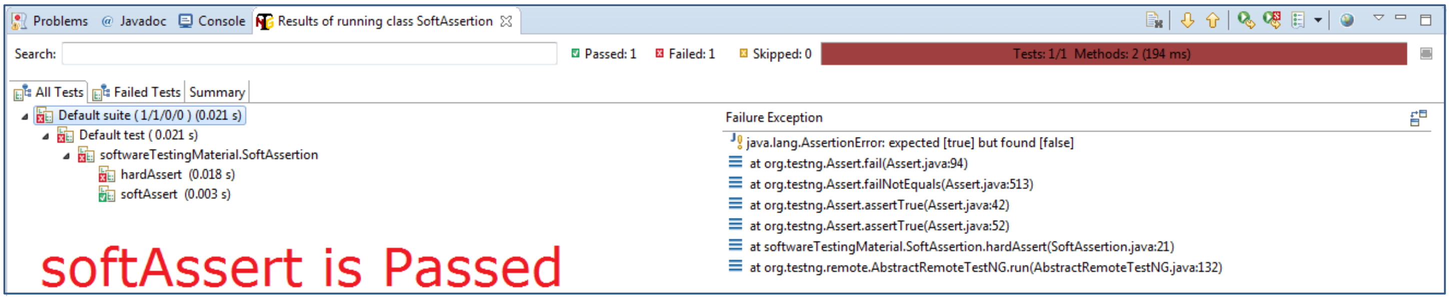
In the ***softAssert()*** method, we have used SoftAssert class and intentionally passing value false in the ***assertTrue***() method to make it fail & In the ***hardAssert()*** method, we simply used Assert and intentionally passing parameter value false in the ***assertTrue***() method to make it fail.

Example Code :

```
public class SoftAssertion {
  @Test
   public void softAssert(){
    SoftAssert softAssertion= new SoftAssert();
    Sopln("softAssert Method Was Started");
    softAssertion.assertTrue(false);
    Sopln("softAssert Method Was Executed");
  }
 @Test
  public void hardAssert(){
     Sopln("hardAssert Method Was Started");
     Assert.assertTrue(false);
     Sopln("hardAssert Method Was Executed");
 } }
```

Console Output :

```
[TestNG] Running:

hardAssert Method Was Started
softAssert Method Was Started
softAssert Method Was Executed

===========================
Default suite
Total tests run: 2, Failures: 1, Skips: 0
==========================
```



Passed: 1 Failed: 1 Skipped: 0 Tests: 1/1 Methods: 2 (194 ms)

```
Default suite (1/1/0/0) (0.021 s)
  Default test (0.021 s)
    softwareTestingMaterial.SoftAssertion
      hardAssert (0.018 s)
      softAssert (0.003 s)
```

Failure Exception

```
java.lang.AssertionError: expected [true] but found [false]
at org.testng.Assert.fail(Assert.java:94)
at org.testng.Assert.failNotEquals(Assert.java:513)
at org.testng.Assert.assertTrue(Assert.java:42)
at org.testng.Assert.assertTrue(Assert.java:52)
at softwareTestingMaterial.SoftAssertion.hardAssert(SoftAssertion.java:21)
at org.testng.remote.AbstractRemoteTestNG.run(AbstractRemoteTestNG.java:132)
```

softAssert is Passed

Below script is same as the first one but just added *assertAll()* method in the end of the first method (i.e.,*softAssert()*).

Note: If you forget to call *assertAll()* at the end of your test, the test **will pass** even if any assert objects threw exceptions as shown in the above example. So don't forget to add *assertAll().*

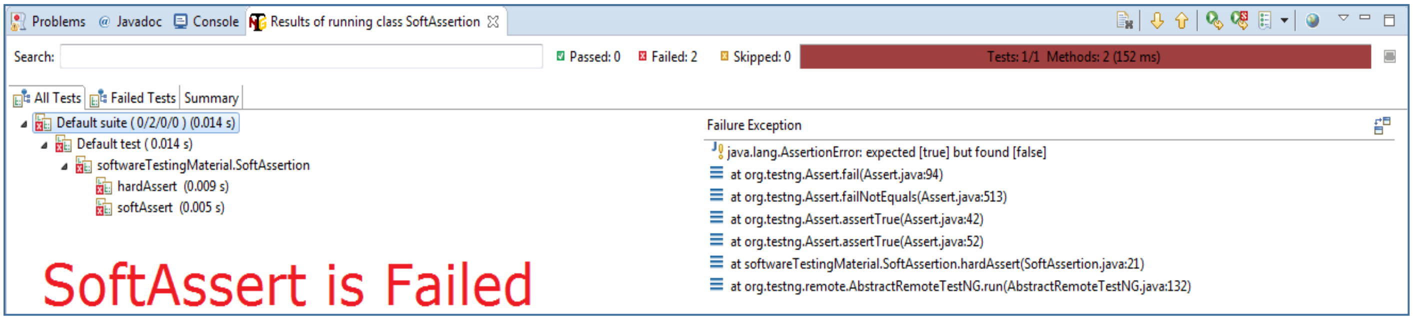
Example Code :

```
public class SoftAssertion {
 @Test
  public void softAssert(){
  SoftAssert softAssertion= new SoftAssert();
  Sopln("softAssert Method Was Started");
  softAssertion.assertTrue(false);
  Sopln("softAssert Method Was Executed");
  softAssertion.assertAll();
 }
  @Test
  public void hardAssert(){
  Sopln("hardAssert Method Was Started");
  Assert.assertTrue(false);
  Sopln("hardAssert Method Was Executed");
}}
```

Console Output:

```
[TestNG] Running:

hardAssert Method Was Started
softAssert Method Was Started
softAssert Method Was Executed

============================
Default suite
Total tests run: 2, Failures: 2, Skips: 0
============================
```

Execute the above script and see the console output. There are two failures here. Second failure is due to *assertAll()*method



Passed: 0 Failed: 2 Skipped: 0 Tests: 1/1 Methods: 2 (152 ms)

```
Default suite (0/2/0/0) (0.014 s)
  Default test (0.014 s)
    softwareTestingMaterial.SoftAssertion
      hardAssert (0.009 s)
      softAssert (0.005 s)
```

Failure Exception

```
java.lang.AssertionError: expected [true] but found [false]
at org.testng.Assert.fail(Assert.java:94)
at org.testng.Assert.failNotEquals(Assert.java:513)
at org.testng.Assert.assertTrue(Assert.java:42)
at org.testng.Assert.assertTrue(Assert.java:52)
at softwareTestingMaterial.SoftAssertion.hardAssert(SoftAssertion.java:21)
at org.testng.remote.AbstractRemoteTestNG.run(AbstractRemoteTestNG.java:132)
```

SoftAssert is Failed

What is the difference between Assert and verify ?

Assert is used to achieve Hard assert whereas verify can be achieved using soft assert class

Eg   Assert.assertequals(“Hi”,”Hello”);

        Assert.assertequals(“Hi”,”Hi”);

Here 2nd statement will not execute because first statement is going to fail.

e.g.   SoftAssert SA = new SoftAssert();

        SA.Assertequals(“Hi”,”Hello”);

        SA.Assertequals(“Hello”,”Hello”);

In above, 2nd statement will also get executed even if 1st statement gets fail.

14. Listeners –

Listeners “listen” to the event defined in the selenium script and behave accordingly. The main purpose of using listeners is to create logs. There are many types of listeners such as ***WebDriver Listeners*** and ***TestNG Listeners***.

Step 1: Create a Class “***ListenerTestNG***” to implement ***ITestListener*** methods

```
public class ListenerTestNG implements ITestListener{
            @Override
            public void onTestStart(ITestResult result) { }
            @Override
            public void onTestSuccess(ITestResult result) {
              Sopln("The name of the testcase passed is :"+Result.getName()");
            }
            @Override
            public void onTestFailure(ITestResult result) {
              Sopln("The name of the testcase failed is :"+Result.getName()");
            }
            @Override
            public void onTestSkipped(ITestResult result) {
                Sopln("The name of the testcase skipped is :"+Result.getName()");
             }
            @Override
            public void onTestFailedButWithinSuccessPercentage(ITestResult result) { }

            @Override
            public void onStart(ITestContext context) { }

            @Override
            public void onFinish(ITestContext context) { }
}
```

Step 2: Create another Class “***ListenerTestNGTestCase***” and write a script (which ever script you prefer). Else copy paste the below mentioned script.

Step 3: Add the listeners annotation (*@Listeners*) in the Class “***ListenerTestNGTestCase***”.The complete “***ListenerTestNGTestCase***” class after adding Listener annotation is mentioned below:

```
@Listeners(listeners.ListenerTestNG.class)
public class ListenerTestNGTestCase {
        WebDriver driver= new FirefoxDriver();
       // Test to pass as to verify listeners .
      @Test(priority=1)
      public void TestToPass()
     {
       System.out.println("This method to pass test");
      driver.get("https://www.softwaretestingmaterial.com/");
      driver.getTitle();
      driver.quit();
     }
    //Used skip exception to skip the test
    @Test(priority=2)
    public void TestToSkip ()
    {
     System.out.println("This method to skip test");
     throw new SkipException("Skipping - This is not ready for testing ");
    }
  // In the above method, we have already closed the browser.
     So we couldnot get the title here. It is to forcefully fail the test
  @Test(priority=3)
   public void TestToFail()
  {  driver.getTitle();
     System.out.println("This method to test fail");
  }}
```

Step 4: Execute the “*ListenerTestNGTestCase*” class. Methods in class “ListenerTestNG” are called automatically according to the behavior of methods annotated as @Test.

Step 5: Verify the Output in the console. You could find the logs in the console. Add the bold lines of code in the TestNG.xml file shown below. –

```
<suite name="Suite">
<listeners>
<listener class-name="listeners.listenerTestNG"/>
</listener>
<test name="Test">
<classes>
   <classname="listeners.ListenerTestNGTestCase">

</classes>
</test>
</suite>
```

Execute it by right clicking on testng.xml and run as TestNG Suite

15. IRetryAnalyzer – How To Run Failed Test Cases –

We could execute the failed test cases in two ways.

Case 1: Execute failed test cases using TestNG in Selenium – By using "testng-failed.xml"

Steps To follow:

1. After the first run of an automated test run. Right click on Project – Click on Refresh
2. A folder will be generated named "***test-output***" folder, Inside "test-output" folder, you could find "***testng-failed.xml***"
3. Run "***testng-failed.xml***" to execute the failed test cases again.

Case 2: Execute failed test cases using TestNG in Selenium – By Implementing TestNG IRetryAnalyzer.

Create a class ***RetryFailedTestCases*** to implement ***IRetryAnalyzer***.

```
public class RetryFailedTestCases implements IRetryAnalyzer {
   private int retryCnt = 0;
   //You could mentioned maxRetryCnt (Maximiun Retry Count) as per your requirement.
   //Here I took 2, If any failed testcases then it runs two times
   private int maxRetryCnt = 2;

   //This method will be called everytime a test fails. It will return TRUE if a test fails
   // and need to be retried, else it returns FALSE
   public boolean retry(ITestResult result) {
      if (retryCnt < maxRetryCnt) {
         System.out.println("Retrying " + result.getName() + " again and the count is " + (retryCnt+1));
         retryCnt++;
         return true;
      }
      return false;

}}
```

Let's create another class 'RetryListenerClass' by Implementing 'IAnnotationTransaformer' interface. transform method is called for every test during test run. A simple implementation of this 'IAnnotationTransformer' interface can help us set the 'setRetryAnalyzer' for 'ITestAnnotation'. Add the above class name (RetryFailedTestCases.class) in the below program. This interface does its work in run time by adding annotation to the test methods.

RetryListenerClass implements IAnnotationTransformer:

```
public class RetryListenerClass implements IAnnotationTransformer {
 @Override
public void transform(ITestAnnotation testannotation, Class testClass, Constructor testConstructor, Method
testMethod) {
   IRetryAnalyzer retry = testannotation.getRetryAnalyzer();
   if (retry == null) {
   testannotation.setRetryAnalyzer(RetryFailedTestCases.class);
   }
}}
```

Let us see the example by executing simple tests below. Here I took two test cases say Test1 and Test2.

Testcase 1:

```
public class Test1 {
            @Test
            public void test1(){
            System.out.println("Test 1");
            Assert.assertTrue(true);
             }
           }
```

Testcase 2:

```
public class Test2 {
            @Test
            public void test2(){
            System.out.println("Test 2");
            Assert.assertTrue(false);
            }         }
```

As per the lines of code in Test2, it will fail. So it (Test2) will be executed 2 times (we took the maxRetryCnt as

2) in Retry Class. First lets include below mentioned Listener to testng.xml file. Below mentioned syntax is to add Listener for RetryListnereClass. Final testng.xml file should looks like below:

```
<suite name="My Suite">
  <listeners>
       <listener class-name="softwareTestingMaterial.RetryListenerClass"/>
 </listeners>
 <test name="Test1">
 <classes>
   <class name="softwareTestingMaterial.Test1" />
 </classes>
 </test> <!-- Test -->
   <test name="Test2">
 <classes>
 <class name="softwareTestingMaterial.Test2" />
 </classes>
 </test> <!-- Test -->

</suite> <!-- Suite -->
```

Execute the testng.xml. Here is the output which I got. You could see in the below mentioned result that the Test 2 is executed three times as we have mentioned ‘maxRetryCnt = 2’. Even though we have just 2 tests, we could find total test runs are 4 in the result.

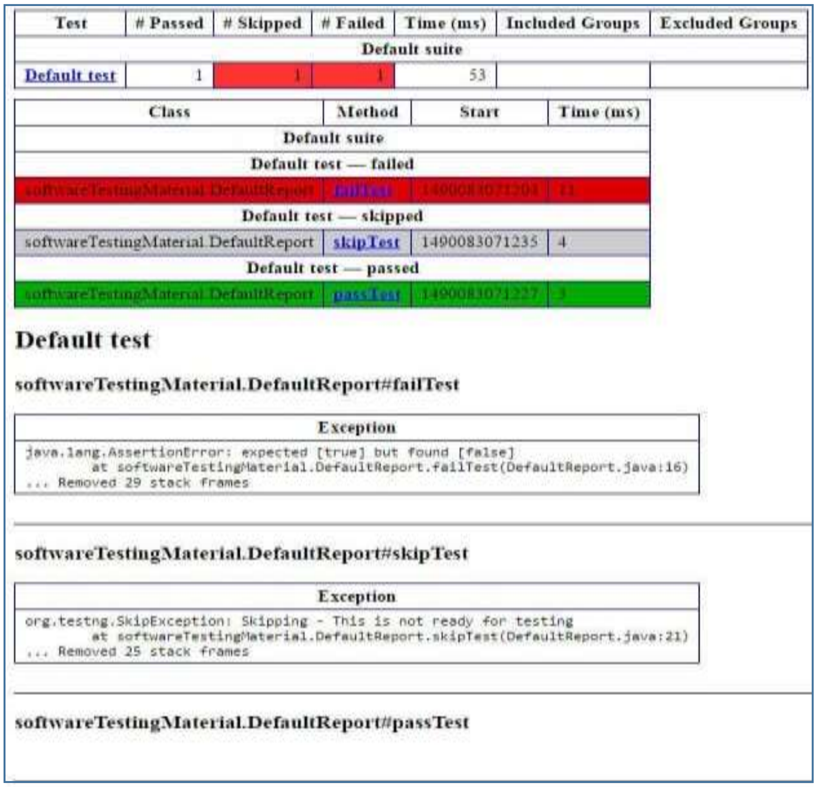
Output On Console -

```
[TestNG] Running:
  Test 1
  Test 2
  Retrying test2 again and the count is 1
  Test 2
  Retrying test2 again and the count is 2
  Test 2

  ===============================
  Everjobs Suite
  Total tests run: 4, Failures: 1, Skips: 2
  ===============================
```

This way we could run failed test cases using TestNG in Selenium.

16. TestNG Reports

Once we execute test cases using TestNG, it will generate a default HTML report. Open in ‘***test-output***’ folder. Now you should find a report ”***emailable-report.html***‘. This is the default report generated by TestNG. Open ***emailable -report.html*** using any browser of your choice.



Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
Default suite						
Default test	1	1	1	53		

Class	Method	Start	Time (ms)
Default suite			
Default test — failed			
softwareTestingMaterial.DefaultReport	failTest	1490083071204	11
Default test — skipped			
softwareTestingMaterial.DefaultReport	skipTest	1490083071235	4
Default test — passed			
softwareTestingMaterial.DefaultReport	passTest	1490083071227	3

Default test

softwareTestingMaterial.DefaultReport#failTest

Exception

```
java.lang.AssertionError: expected [true] but found [false]
        at softwareTestingMaterial.DefaultReport.failTest(DefaultReport.java:16)
... Removed 29 stack frames
```

softwareTestingMaterial.DefaultReport#skipTest

Exception

```
org.testng.SkipException: Skipping - This is not ready for testing
        at softwareTestingMaterial.DefaultReport.skipTest(DefaultReport.java:21)
... Removed 25 stack frames
```

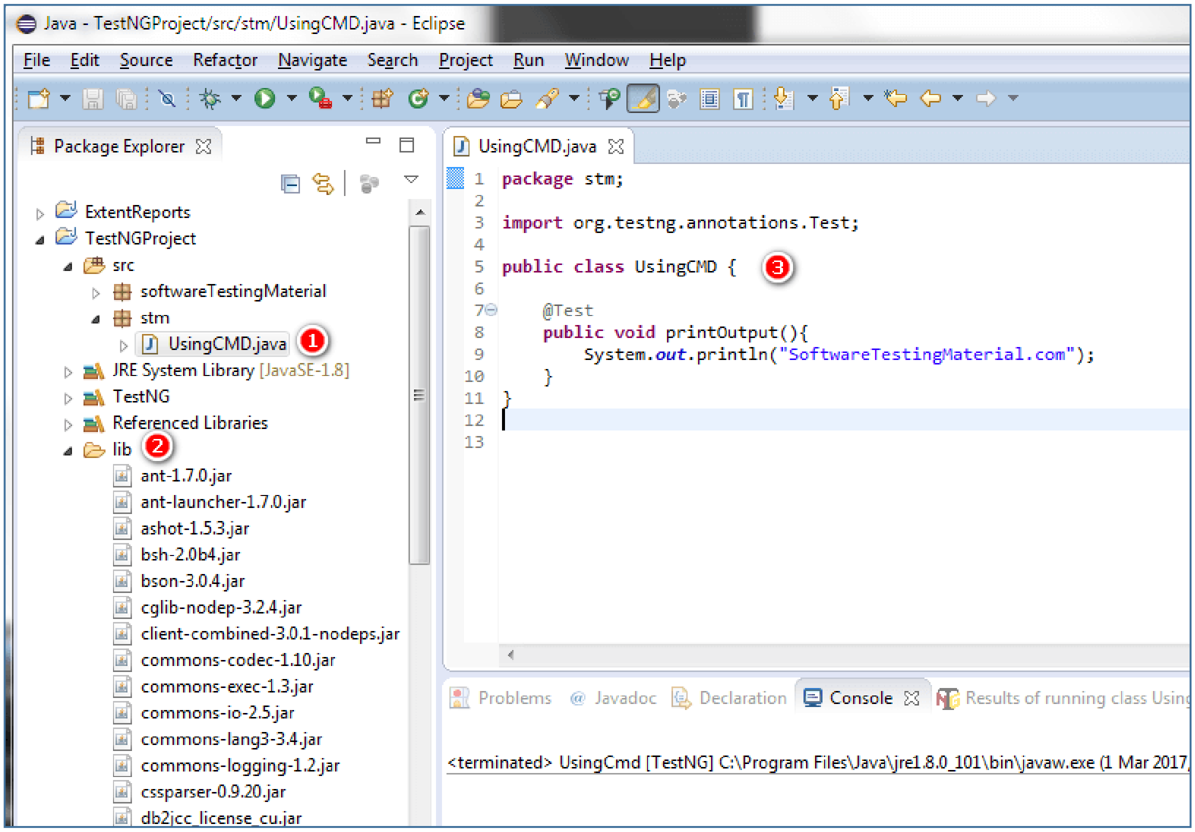
softwareTestingMaterial.DefaultReport#passTest

17. How To Run TestNG Using Command Prompt ?

Step i: Open Eclipse and create a Java class

Step ii. Keep all the library files in a folder (here I create a folder name “lib”)

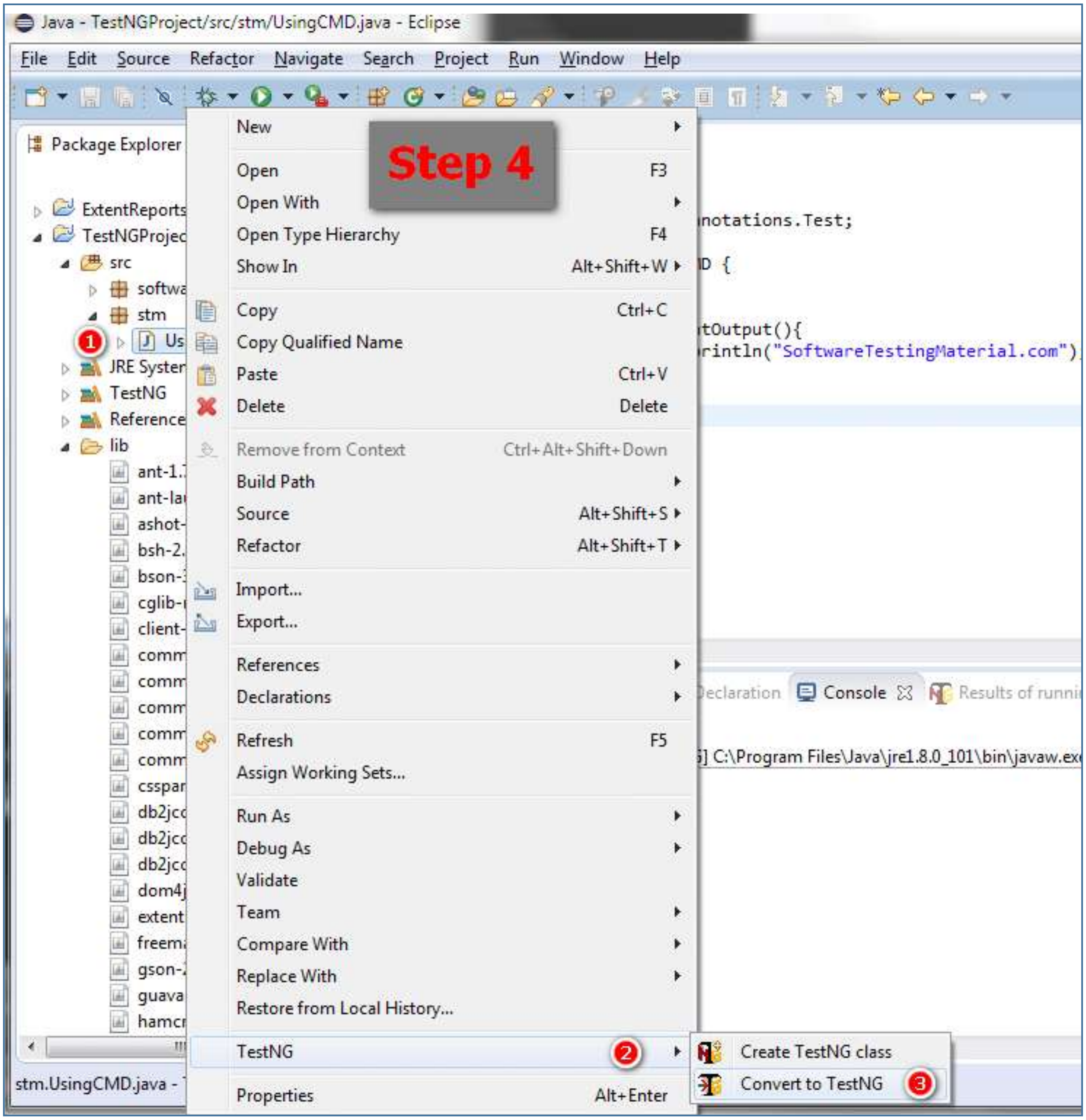
Step iii: Write a Java program



```java
package stm;

import org.testng.annotations.Test;

public class UsingCMD {

    @Test
    public void printOutput(){
        System.out.println("SoftwareTestingMaterial.com");
    }
}
```

```
<terminated> UsingCmd [TestNG] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (1 Mar 2017,
```

Step iv: Convert the Java Program into TestNG



Step v: Open command prompt

Step vi: Run the TestNG using command prompt

```
C:\Users\workspace\SoftwareTestingMaterial
set classpath=C:\workspace\SoftwareTestingMaterial\bin; C:\ workspace\SoftwareTestingMaterial\lib\*
java org.testng.TestNG C:\workspace\SoftwareTestingMaterial\testng.xml
```

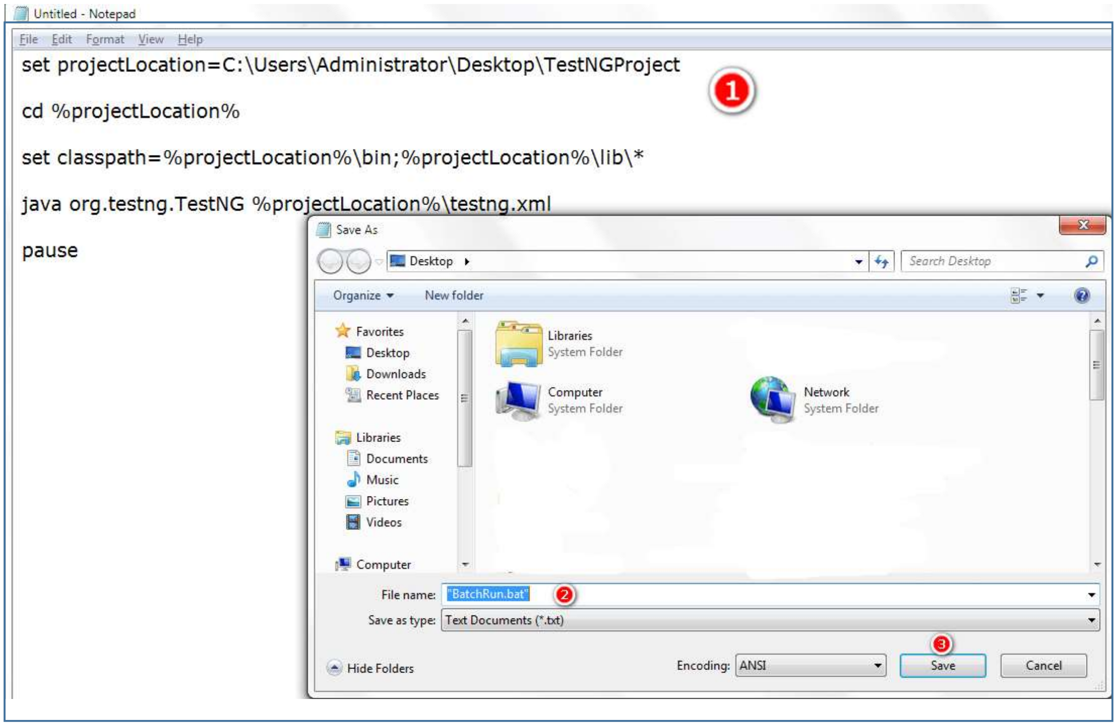
Also you could run TestNG using Batch file (.bat file)

Copy the below code and place it in a notepad and save the file using .bat extension

set projectLocation =

C:\Users\Admin\Desktop\STMSeleniumTutorial\workspace\SoftwareTestingMaterial

```
cd %projectLocation%
set classpath=%projectLocation%\bin;%projectLocation%\lib\*
java org.testng.TestNG %projectLocation%\testng.xml
pause
```

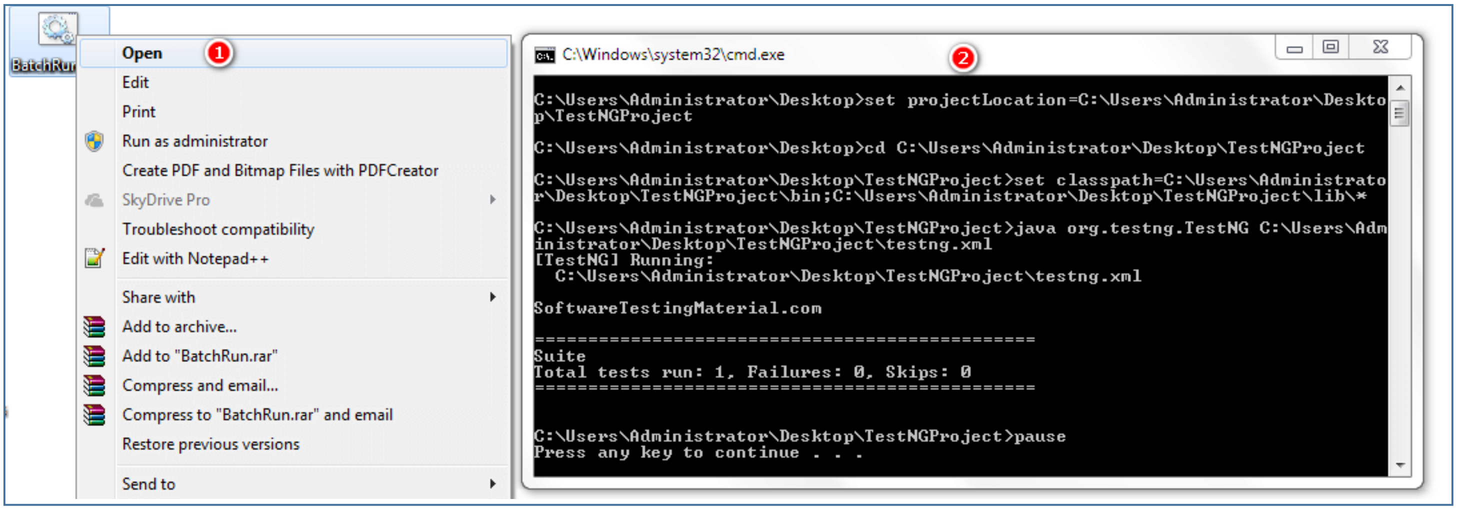


```
set projectLocation=C:\Users\Administrator\Desktop\TestNGProject

cd %projectLocation%

set classpath=%projectLocation%\bin;%projectLocation%\lib\*

java org.testng.TestNG %projectLocation%\testng.xml

pause
```

File name: "BatchRun.bat"

Save as type: Text Documents (*.txt)

Encoding: ANSI



BatchRun
Open
1
Edit
Print
Run as administrator
Create PDF and Bitmap Files with PDFCreator
SkyDrive Pro
Troubleshoot compatibility
Edit with Notepad++
Share with
Add to archive...
Add to "BatchRun.rar"
Compress and email...
Compress to "BatchRun.rar" and email
Restore previous versions
Send to
C:\Windows\system32\cmd.exe
2

```
C:\Users\Administrator\Desktop>set projectLocation=C:\Users\Administrator\Deskto
p\TestNGProject

C:\Users\Administrator\Desktop>cd C:\Users\Administrator\Desktop\TestNGProject

C:\Users\Administrator\Desktop\TestNGProject>set classpath=C:\Users\Administrato
r\Desktop\TestNGProject\bin;C:\Users\Administrator\Desktop\TestNGProject\lib\*

C:\Users\Administrator\Desktop\TestNGProject>java org.testng.TestNG C:\Users\Adm
inistrator\Desktop\TestNGProject\testng.xml
[TestNG] Running:
  C:\Users\Administrator\Desktop\TestNGProject\testng.xml

SoftwareTestingMaterial.com

===============================================
Suite
Total tests run: 1, Failures: 0, Skips: 0
===============================================


C:\Users\Administrator\Desktop\TestNGProject>pause
Press any key to continue . . .
```

18. TestNG Reporter Logs

We need the information which helps the User to understand the test steps or any failure during the test case execution. With the help of ***TestNG Reporter Logs*** it is possible to enable logging during the Selenium test case execution.

In selenium there are two types of logging. **High level** logging and **Low level** logging. In low level logging we try to produce logs for the every step you take or every action you make in your automation script. In high level logging you just try to capture main events of your test.

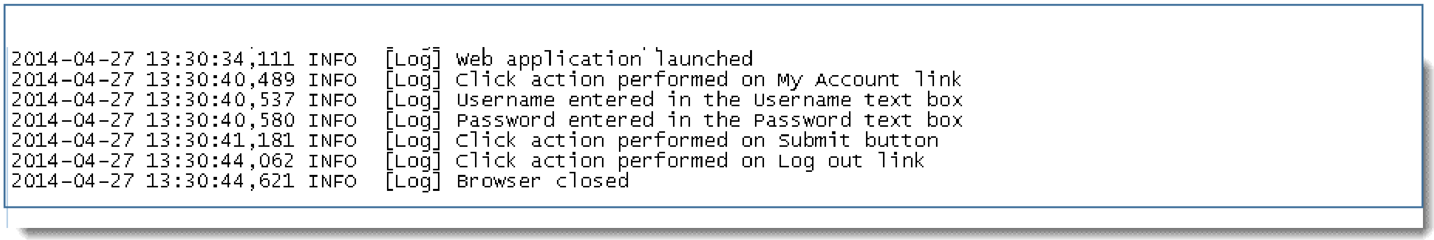
Here, perform **low level logging with *log4j*** and **high level logging with *Testng reporter logs***.

1) Write a test case for Sign In application and implement Log4j logging on every step.

2) Insert Reporter logs on the main events of the test.

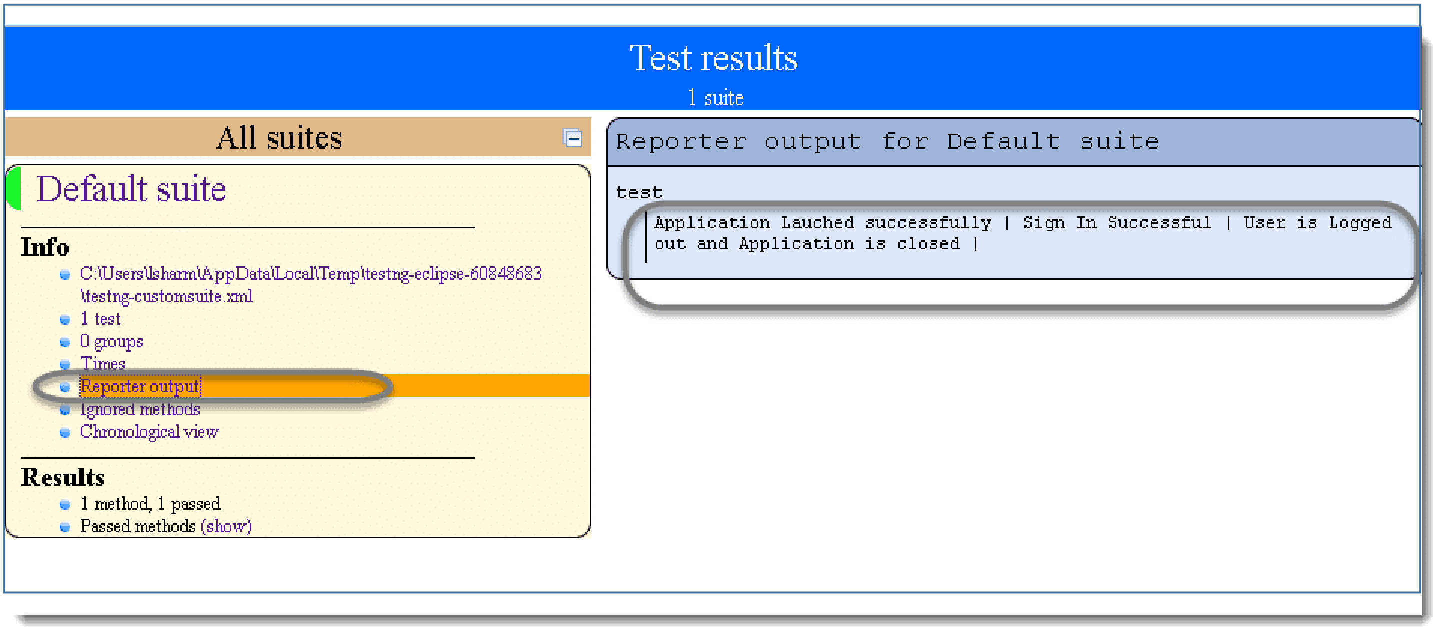
```
public class ReporterLogs {
     public static WebDriver driver;
     public static Logger Log = Logger.getLogger(Log.class.getName())
     @Test
     public static void test() {
            DOMConfigurator.configure("log4j.xml");
            driver = new FirefoxDriver();
            Log.info("New driver instantiated");
            driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
            Log.info("Implicit wait applied on the driver for 10 seconds");
            driver.get("http://www.store.demoqa.com");
            Log.info("Web application launched");
            Reporter.log("Application Lauched successfully | ");                // First high level Event
             driver.findElement(By.xpath(".//*[@id='account']/a")).click();
             Log.info("Click action performed on My Account link");
            driver.findElement(By.id("log")).sendKeys("testuser_1");
             Log.info("Username entered in the Username text box");
            driver.findElement(By.id("pwd")).sendKeys("Test@123");
             Log.info("Password entered in the Password text box");
             driver.findElement(By.id("login")).click();
             Log.info("Click action performed on Submit button");
             Reporter.log("Sign In Successful | " );                            // Second High level event
             driver.findElement(By.id("account_logout"));
             Log.info("Click action performed on Log out link");
             driver.quit();
             Log.info("Browser closed");
            Reporter.log("User is Logged out and Application is closed | "); // Third high level Event
    } }
```

Run the Test Script ***Run as TestNG,*** and check following Log file will be generated in logs folder



```
2014-04-27 13:30:34,111 INFO  [Log] Web application launched
2014-04-27 13:30:40,489 INFO  [Log] Click action performed on My Account link
2014-04-27 13:30:40,537 INFO  [Log] Username entered in the Username text box
2014-04-27 13:30:40,580 INFO  [Log] Password entered in the Password text box
2014-04-27 13:30:41,181 INFO  [Log] Click action performed on Submit button
2014-04-27 13:30:44,062 INFO  [Log] Click action performed on Log out link
2014-04-27 13:30:44,621 INFO  [Log] Browser closed
```

Check the index.html file in output folder of TestNG, reporters log will look like this -



Test results

1 suite

All suites

Default suite

Info

- C:\Users\lsharm\AppData\Local\Temp\testng-eclipse-60848683\testng-customsuite.xml
- 1 test
- 0 groups
- Times
- Reporter output
- Ignored methods
- Chronological view

Results

- 1 method, 1 passed
- Passed methods (show)

Reporter output for Default suite

```
test
Application Lauched successfully | Sign In Successful | User is Logged
out and Application is closed |
```

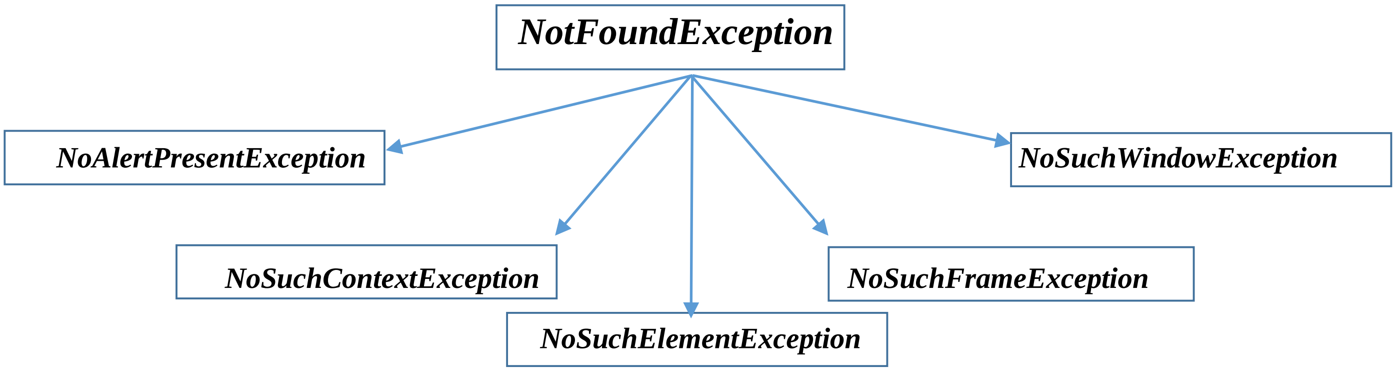
Log4j logging will help you to report a bug or steps taken during the test, on the other hand reporters log will help you to share the test status with leadership. As leadership is just interested in the test results, not the test steps.

We can use reporter's logs on the verification during the test. For example

```
if(Text1.equals(Text2)){

 Reporter.log("Verification Passed forText");

 }else{

 Reporter.log("Verification Failed for Text");

 }
```

Stale Element Ref Exception & No Such Element Exception

Hierarchy :



The ***NotFoundException*** is a super class which includes the subclass ***NoSuchElementException***. The known direct subclasses of ***NotFoundException*** are: ***NoAlertPresentException***, ***NoSuchContextException***, ***NoSuchElementException***, ***NoSuchFrameException*** and ***NoSuchWindowException***.

- **Difference between *NoSuchElementException, StaleElelmentException* –**

1. The ***NoSuchElementException*** is thrown when the element you are trying to find a Element which is not in the DOM. This can happen for three reasons.

   A. When element does not exist and never will, To fix this, change your findElement to be correct.
   B. When element does not exist until you will do some actions, means until you are not selecting a country, list of metro cities of that specific country will not be visible on DOM.
   C. The third is that the element is generated by javascript but WebDriver attempts to find the element before the javascript has created it. The fix for this is to use WebDriverWait to wait for the element to appear (visibility and / or clickable).

2. ***StaleElementReferenceException*** is when you find an element, the DOM gets modified then you reference the WebElement. For example,

```
WebElement we = driver.findElement(By.cssSelector("#valid"));
// you do something which alters the page or a javascript event alters the page
we.click();
```

A classic example if this might be:

```
List<WebElement> listOfAnchors = driver.findElements(By.tag("a"));
for(WebElement anchor : listOfAnchors) {
anchor.click();
System.out.println(driver.getTitle());
driver.navigate.back();
}
```

This code will get all the anchor elements into a list. Lets say there are 5 anchors on the page. The list now has 5 WebElement references. We get the first reference and click it. This take us to a new page. This is a new DOM. We print the title of the new page. Then we use ***back( )*** to go back to the original page. The DOM looks just like the same DOM but it is a different DOM. So now all the references in the list are stale. On the second iteration, it gets the second reference and clicks it. This will throw a ***StaleElementReferenceException***.

More difficult to debug is:

```
WebElement we = driver.findElement(...);
// javascript event gets fired by the website
we.click();
```

Sometimes this will throw a ***StaleElementReferenceException*** but sometimes the timing will be different and the click will work. I've seen many people have this intermittent problem. They add more code which doesn't fix the problem. The extra code just changes the timing and hides the problem. The problem comes back a few days later. So they add more random code. It looks like they fixed the problem but they just changed the timing. So if you get a ***StaleElementReferenceException*** and it is not clear why, it is probably this problem and you need to figure out how to make the findElement and click atomic.

WebDriver Event Listener (Notes By – Vaibhav Mhaskar)

Listeners "listen" to the event defined in the selenium script and behave accordingly. The main purpose of using listeners is to create logs and reports.

WebDriverEventListener :

This ***WebDriverEventListener*** interface allows us to implement the methods. While Test script is executing, Selenium WebDriver performs activities such as Type, Click, Navigate etc., To keep track of these activities we can use WebDriver Event Listeners interface.

EventFiringWebDriver :

This ***EventFiringWebDriver*** class actually fire WebDriver event. Lets see how to implement Listeners in Selenium WebDriver Script.

Step 1: Create a Class "***EventCapture***" to implement ***WebDriverEventListener*** methods
Step 2: Create another Class "***ListenerMainClass***" and write a Test script (which ever script you prefer)
Step 3: In the Class "***ListenerMainClass***", Create ***EventFiringWebDriver*** object and pass driver object as a parameter.

```
EventFiringWebDriver  eventHandler  =  new EventFiringWebDriver (driver);
```

Step 4: In the Class "***ListenerMainClass***", Create an object of the Class "***EventCapture***" where we implemented all the methods of ***WebDriverEventListener*** *to register with* ***EventFiringWebDriver***.

```
EventCapture  eCapture  =  new  EventCapture();
```

Lets see, each step in detail to implement the WebDriver Event Listeners in Selenium:

Step 1: Create a Class "***EventCapture***" to implement ***WebDriverEventListener*** methods,

```
public class EventCapture implements WebDriverEventListener{
        @Override
          public void afterChangeValueOf(WebElement arg0, WebDriver arg1) {     }
          @Override
          public void afterClickOn(WebElement arg0, WebDriver arg1) {   }
          . . . . . . .  . .
         @Override
          public void afterNavigateBack(WebDriver arg0) { // implementing this method
              Sopln("After Navigating Back "+arg0.getCurrentUrl());
          }
         @Override
          public void beforeNavigateBack(WebDriver arg0) { // implementing this method
               Sopln("Before Navigating Back "+arg0.getCurrentUrl());
          }
  . . .   }
```

Step 2: Create another Class “***ListenerMainClass***” and write a script (which ever script you prefer). Below script opens two webpages one after other and navigates backs to the first webpage.

Step 3: In the Class “***ListenerMainClass***“, Create ***EventFiringWebDriver*** object and pass driver object as a parameter.

Step 4: In the Class “***ListenerMainClass***“, Create an object of the Class “***EventCapture***” where we implemented all the methods of ***WebDriverEventListener*** to register with ***EventFiringWebDriver.***

```
public class ListenerMainClass {
   public static void main (String [] args){
          WebDriver driver = new FirefoxDriver();
          EventFiringWebDriver eventHandler = new EventFiringWebDriver(driver);
          EventCapture eCapture = new EventCapture();
         //Register method allows to register our implementation of WebDriverEventListner to listen to the
          // WebDriver Event
          eventHandler.register(eCapture);    //Registering with EventFiringWebDriver
          eventHandler.navigate().to("https://www.softwaretestingmaterial.com/100-software-testing-questions/");
          eventHandler.findElement(By.id("email")).sendKeys("asdsadsa");
          eventHandler.navigate().to("https://www.softwaretestingmaterial.com/agile-testing-questions/");
          eventHandler.navigate().back();   //navigating back to the first page
          eventHandler.quit();
          eventHandler.unregister(eCapture); //Unregister allows to detach
          System.out.println("End of Listners Class");
  }
}
```

Finally, execute the “***ListenMainClass***” class. Methods in class “EventCapture” are called automatically based on the script. Verify the Output in the console. You could find the logs in the console. The two url’s in the console are just because the implementation of ***beforeNavigateBack( )*** and ***afterNavigateBack( )*** methods**.**

Diffeerence Between WebDriver Listeners & TestNG Listeners –

Listeners are those which will listen to event.

In case of ***WebDriver***, whenever anyone clicks on a button on a web page, then automatically a method will be called ***actionPerformed***() in ActionListener case.

But in case of ***TestNG***, there are the actual "listeners" which listen (here the method gets called automatically) to the ***test execution events***. A few examples are: ***onStart***(), ***beforeStart***(), ***afterFinish***(), and ***onFinish***() etc. Mostly, the ***TestNG*** automation teams implement their own custom listeners for custom Logging and reporting.

WebDriver listeners too do a similar job...of logging and/or reporting. But then both of them work on different event sets. ***WebDriver works on different automation events whereas TestNG works on different test's related events.*** The point to note is that, with WebDriver listeners, "Logging" happens before/after event occurance such as click/SendKeys etc.

*"*WebDriver Listeners are ***Action Event Level Listeners*** & TestNG Listener is ***Test Event Listener"***

How to deal with Captcha ? (Notes By – Vaibhav Mhaskar)

We cannot automate Captcha as it is not meant to be automatized. Suppose we have to automate the test case for the web page which having captcha, so in this scenario let's see how to deal with Captcha.

Option 1 : Either ask development team for a workaround, like configure Captcha in test Environment in such way it will always accept 1 specific value.

Option 2 : Ask developer to disable the Captcha module in Testing environment**.**

Option 3 : If you are using custom captcha module, you can ask developer to generate an API of captcha generation for Testing environment.

Option 4 : You can ask to development team to add captcha code as title in markup then you can access this title and bypass the captcha, but only in the testing environment.

Most of the time we go with option 1 & 2 .

Note : ***When you do work around for captcha in test environment , you should always have this point in your checklist to make sure workarounds foe captcha testing are removed before it goes on production***

MANTIS Bug Tracker Tutorial For Beginners (source – guru99) (Notes By – Vaibhav Mhaskar)

MANTIS is an open source bug tracking software. It can be used to track bug for various software projects. You can easily download and install the Mantis for your use. Mantisbt now also provides a hosted version of the software. Some salient features of Mantis Bt are as follows

- Email notifications: It sends out emails of updates, comments, resolutions to the concerned stakeholders.
- ***Access Control:*** You can control user access at a project level
- ***Customize:*** You can easily customize Mantis as per your requirements.
- ***Mobile Support:*** Mantis supports iPhone, Android, and Windows Phone Platforms.
- ***Plugins:*** An ever-expanding library of plugins to add custom functionality to Mantis Issue Tracker.

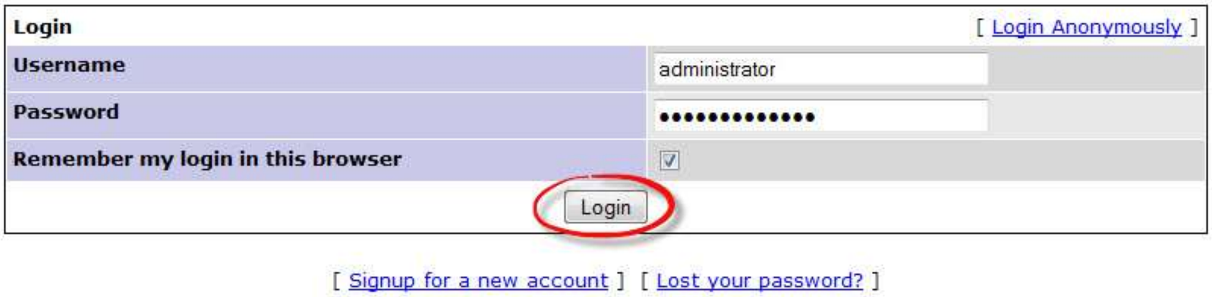
In this tutorial, you will learn -

- How to Report an Issue
- How to Add a Project
- How to Create a User
- How to Create Custom Field
-

How to Report an Issue

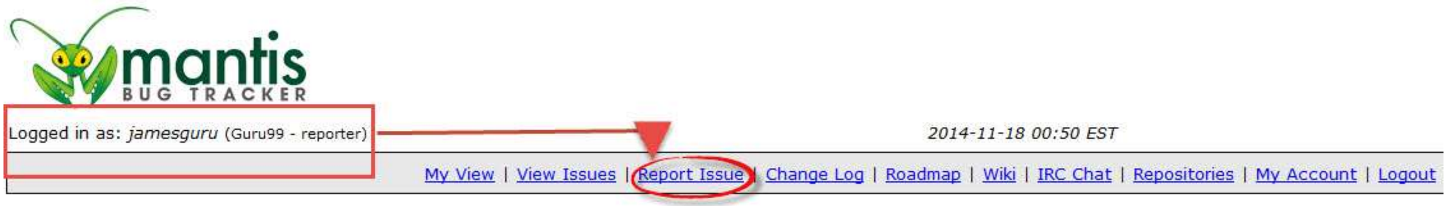
There is no need to download Mantis, you use the online demo

Step 1) Log into Mantis



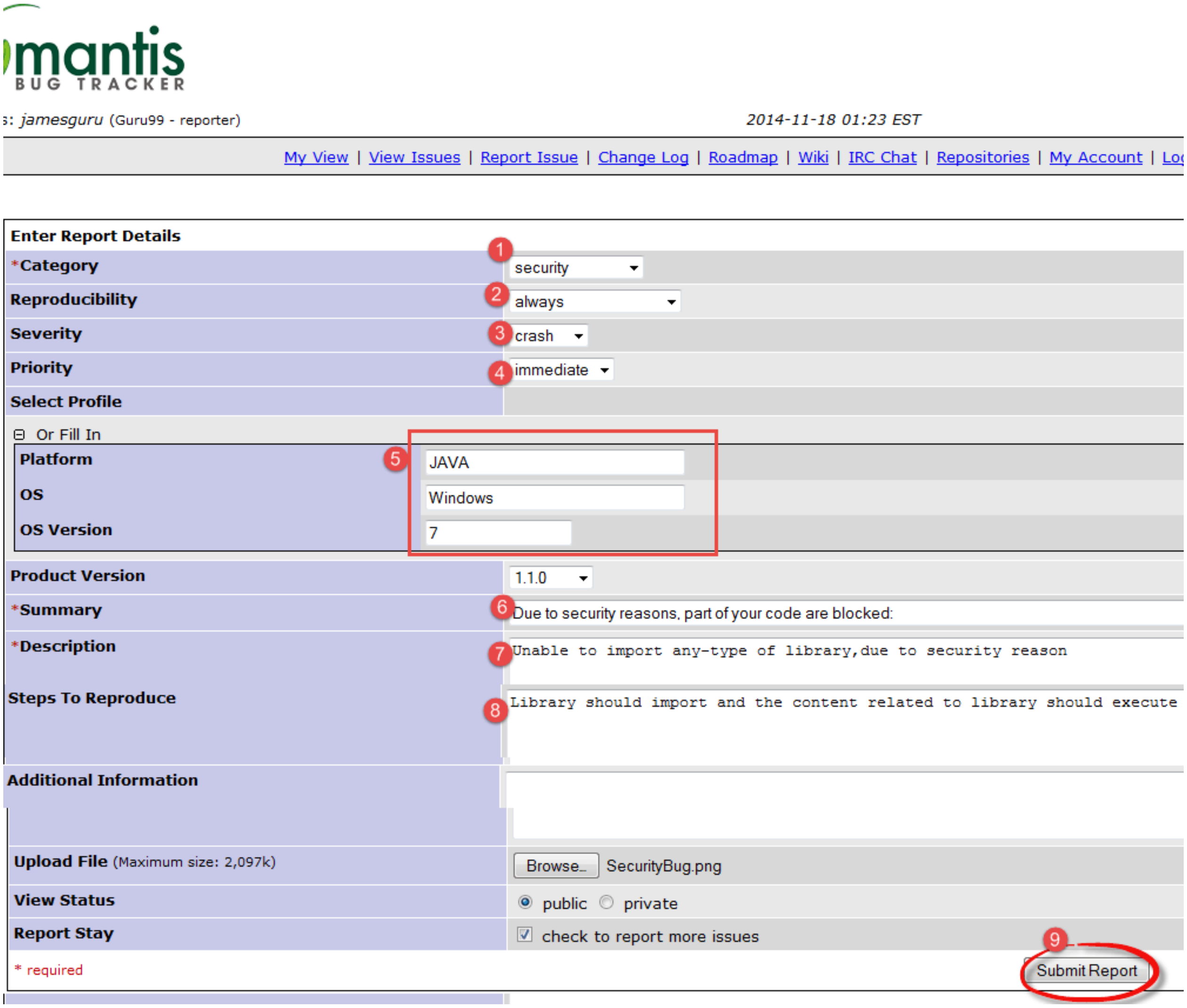
Login	[Login Anonymously]
Username	administrator
Password	•••••••••••••
Remember my login in this browser	☑

Step 2) Once you login to Mantis your user-name will be displayed on the top of the Mantis main screen and now you can report your issue into the Mantis by clicking on the option **"Report Issue"** as shown below.


mantis BUG TRACKER
Logged in as: jamesguru (Guru99 - reporter)
2014-11-18 00:50 EST
My View | View Issues | Report Issue | Change Log | Roadmap | Wiki | IRC Chat | Repositories | My Account | Logout

Step 3) In the next Screen

1. Enter ***Bug Category***
2. Enter ***Reproducibility***
3. Enter ***Severity***
4. Enter ***Priority***
5. Enter ***Platform*** Details
6. Enter Summary of the ***Bug Report***
7. Enter ***Description***
8. Enter Steps to ***reproduce*** the error
9. Click Submit ***Report***

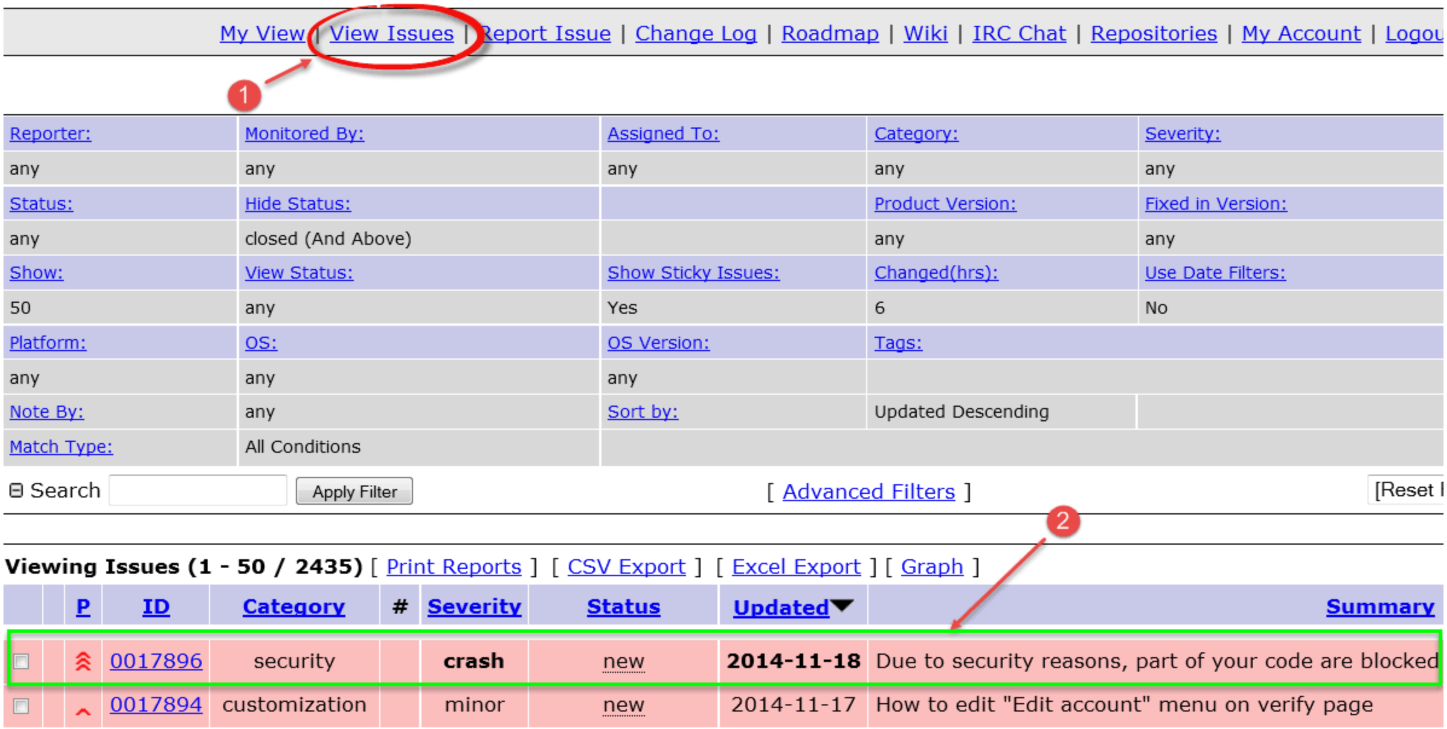

mantis BUG TRACKER
s: jamesguru (Guru99 - reporter)
2014-11-18 01:23 EST
My View | View Issues | Report Issue | Change Log | Roadmap | Wiki | IRC Chat | Repositories | My Account | Lo

Enter Report Details

*Category	1	security
Reproducibility	2	always
Severity	3	crash
Priority	4	immediate
Select Profile		
⊟ Or Fill In		
Platform	5	JAVA
OS		Windows
OS Version		7
Product Version		1.1.0
*Summary	6	Due to security reasons, part of your code are blocked:
*Description	7	Unable to import any-type of library,due to security reason
Steps To Reproduce	8	Library should import and the content related to library should execute
Additional Information		
Upload File (Maximum size: 2,097k)		Browse... SecurityBug.png
View Status		● public ○ private
Report Stay		☑ check to report more issues
* required	9	Submit Report

NOTE: The fields you see in your version of Mantis will defer. Refer our section on Custom Fields for the same.

Step 4) After entering all the details in the report window, we will submit a report and soon our report will be displayed on the main window. As show in the screen-shot below, when you click on view issue, the issue will appear on the screen with the id **"0017896"** also, it shows the status as new and also the date when it was created.



My View | View Issues | Report Issue | Change Log | Roadmap | Wiki | IRC Chat | Repositories | My Account | Logou

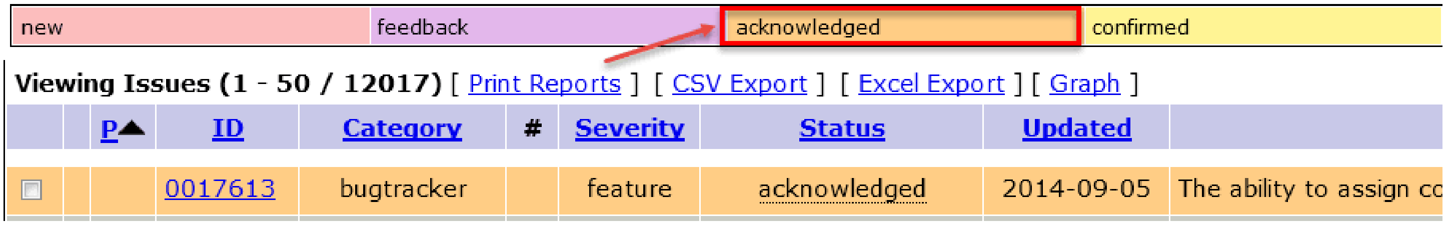
Reporter:	Monitored By:	Assigned To:	Category:	Severity:
any	any	any	any	any
Status:	Hide Status:		Product Version:	Fixed in Version:
any	closed (And Above)		any	any
Show:	View Status:	Show Sticky Issues:	Changed(hrs):	Use Date Filters:
50	any	Yes	6	No
Platform:	OS:	OS Version:	Tags:	
any	any	any		
Note By:	any	Sort by:	Updated Descending	
Match Type:	All Conditions			

⊟ Search [Apply Filter] [Advanced Filters] [Reset I

Viewing Issues (1 - 50 / 2435) [Print Reports] [CSV Export] [Excel Export] [Graph]

	P	ID	Category	#	Severity	Status	Updated▼	Summary
□	⇈	0017896	security		**crash**	new	**2014-11-18**	Due to security reasons, part of your code are blocked
□	^	0017894	customization		minor	new	2014-11-17	How to edit "Edit account" menu on verify page

The issues in the Mantis Tool are highlighted according to the status of an issue. For example here the issue is in acknowledged status, therefore, highlighted in light orange.



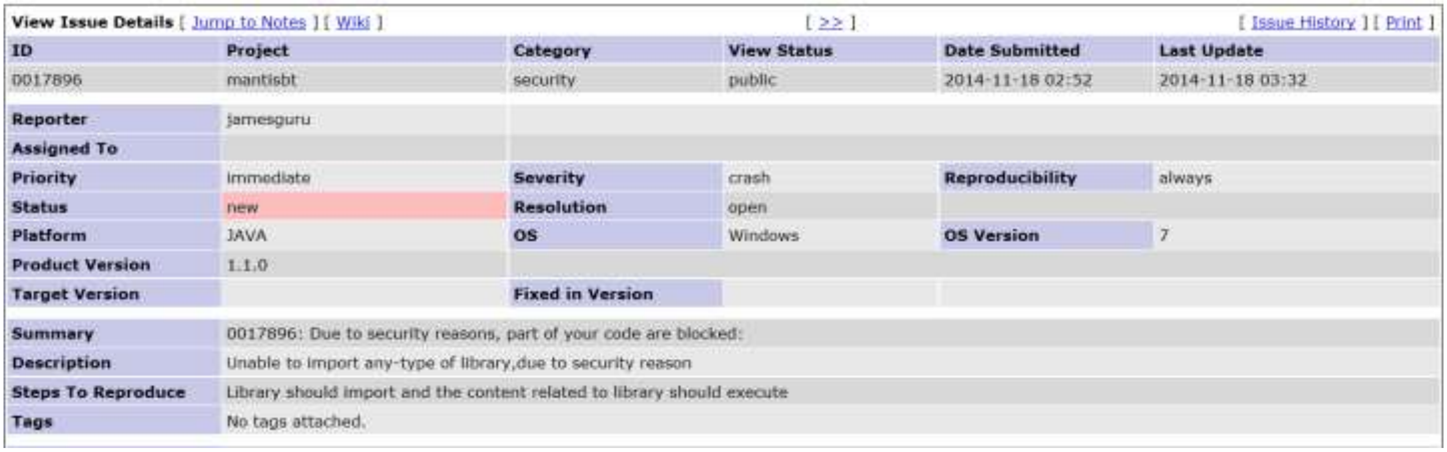
new	feedback	acknowledged	confirmed

Viewing Issues (1 - 50 / 12017) [Print Reports] [CSV Export] [Excel Export] [Graph]

	P▲	ID	Category	#	Severity	Status	Updated	
□		0017613	bugtracker		feature	acknowledged	2014-09-05	The ability to assign cc

For different statuses, different colors are allotted.

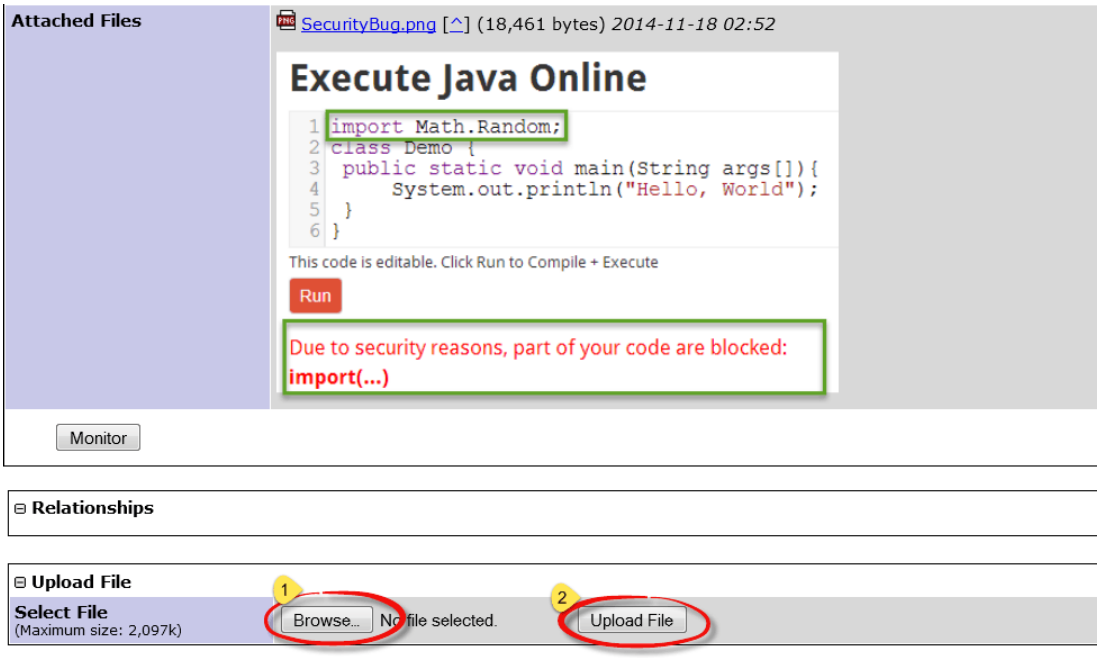
Step 5) When you click on your issue #id **0017896**, it will show the issue in more details like project reporter, its status, date submitted and last updated.



View Issue Details [Jump to Notes] [Wiki] [>>] [Issue History] [Print]

ID	Project	Category	View Status	Date Submitted	Last Update
0017896	mantisbt	security	public	2014-11-18 02:52	2014-11-18 03:32
Reporter	jamesguru				
Assigned To					
Priority	immediate	Severity	crash	Reproducibility	always
Status	new	Resolution	open		
Platform	JAVA	OS	Windows	OS Version	7
Product Version	1.1.0				
Target Version		Fixed in Version			
Summary	0017896: Due to security reasons, part of your code are blocked:				
Description	Unable to import any-type of library,due to security reason				
Steps To Reproduce	Library should import and the content related to library should execute				
Tags	No tags attached.				

Step 6) Apart from these, you can add a file, documents, images to your bug as shown below –



Attached Files

SecurityBug.png [^] (18,461 bytes) *2014-11-18 02:52*

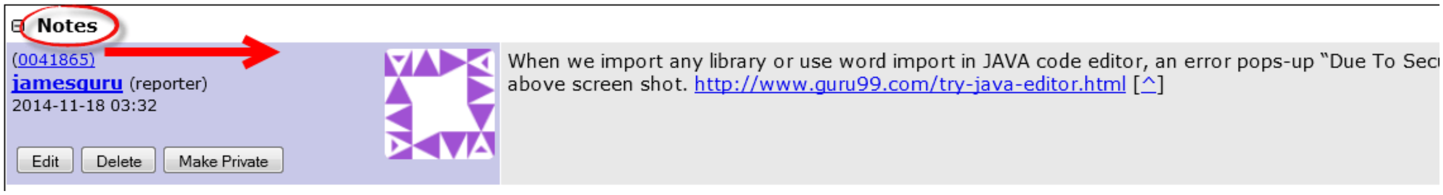
Execute Java Online

```
import Math.Random;
class Demo {
 public static void main(String args[]){
     System.out.println("Hello, World");
 }
}
```

This code is editable. Click Run to Compile + Execute

Due to security reasons, part of your code are blocked:
import(...)

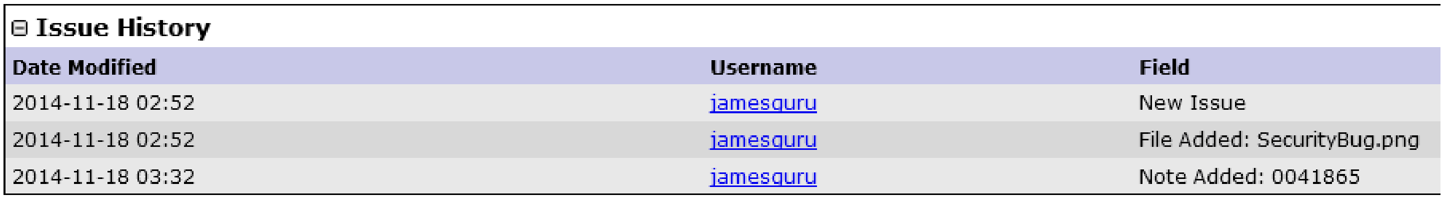
Step 7) You can also add comments addressing your issue as shown in the screen-shot below.



(0041865)
jamesguru (reporter)
2014-11-18 03:32

When we import any library or use word import in JAVA code editor, an error pops-up "Due To Sec
above screen shot. http://www.guru99.com/try-java-editor.html [^]

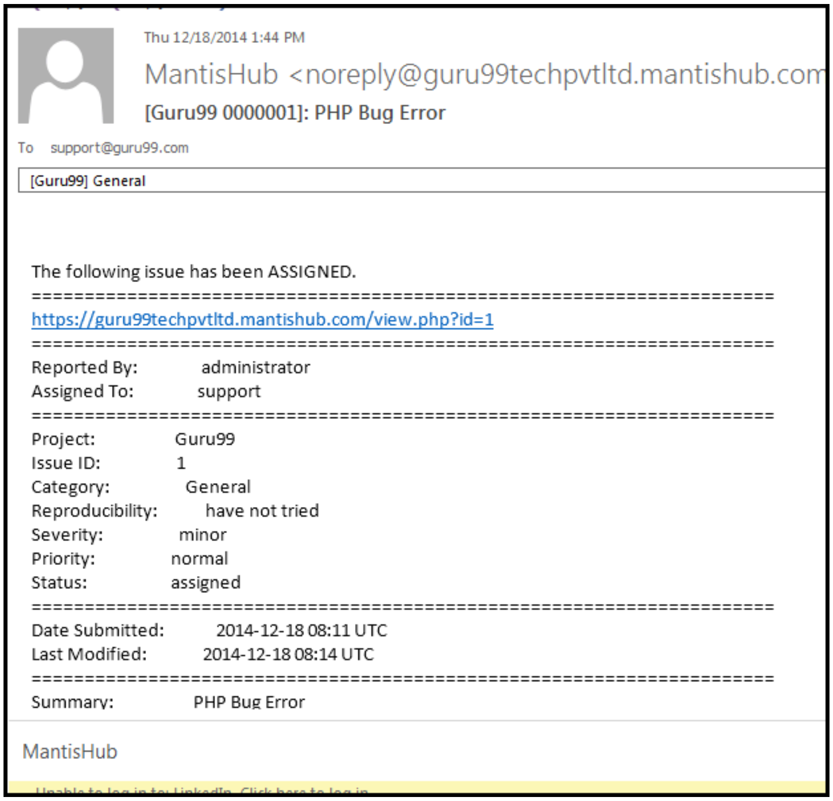
Step 8) You can click history at top of the Issue Report, to see issue history. This issue history has details like when the issue was created, or when the file was added to the issue or if there is any note added to the issue.



⊟ **Issue History**

Date Modified	Username	Field
2014-11-18 02:52	jamesguru	New Issue
2014-11-18 02:52	jamesguru	File Added: SecurityBug.png
2014-11-18 03:32	jamesguru	Note Added: 0041865

Step 9) Once the issue is submitted the developer receives an email.



Thu 12/18/2014 1:44 PM

MantisHub <noreply@guru99techpvtltd.mantishub.com

[Guru99 0000001]: PHP Bug Error

To support@guru99.com

[Guru99] General

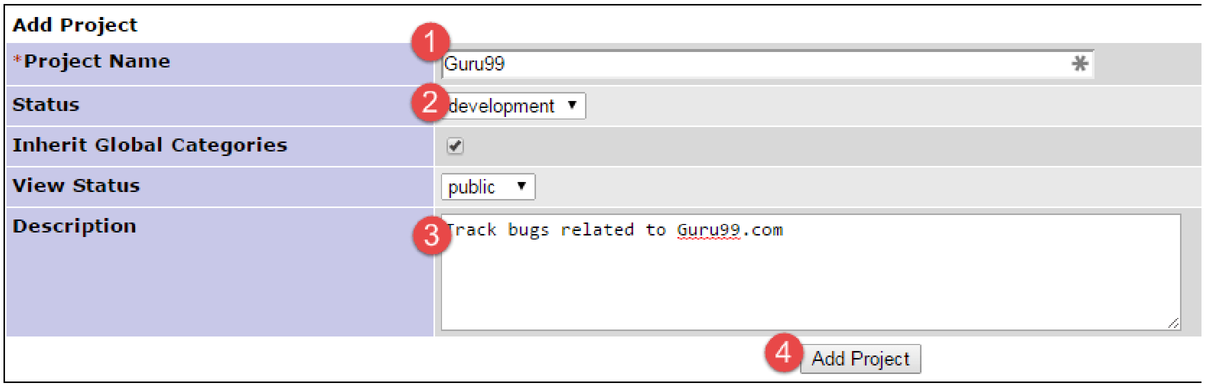
```
The following issue has been ASSIGNED.
======================================================================
https://guru99techpvtltd.mantishub.com/view.php?id=1
======================================================================
Reported By:          administrator
Assigned To:          support
======================================================================
Project:              Guru99
Issue ID:             1
Category:             General
Reproducibility:      have not tried
Severity:             minor
Priority:             normal
Status:               assigned
======================================================================
Date Submitted:       2014-12-18 08:11 UTC
Last Modified:        2014-12-18 08:14 UTC
======================================================================
Summary:              PHP Bug Error
```

MantisHub

How to Add a Project

Step 1) Under Manage Tab, go to Manage Projects

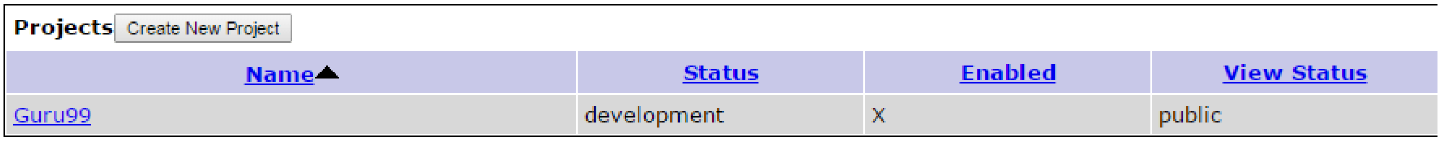
1. Enter Project Name
2. Select Status
3. Enter Description
4. Click Add Project



Add Project

*Project Name	Guru99
Status	development ▼
Inherit Global Categories	☑
View Status	public ▼
Description	Track bugs related to Guru99.com

Step 2) Project is created.

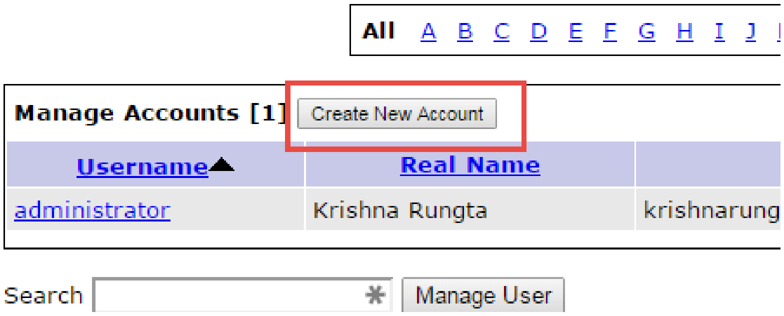


Projects

Name▲	Status	Enabled	View Status
Guru99	development	X	public

How to Create a User

Step 1) Go to Manage > Manage Users. Click "Create New Account"

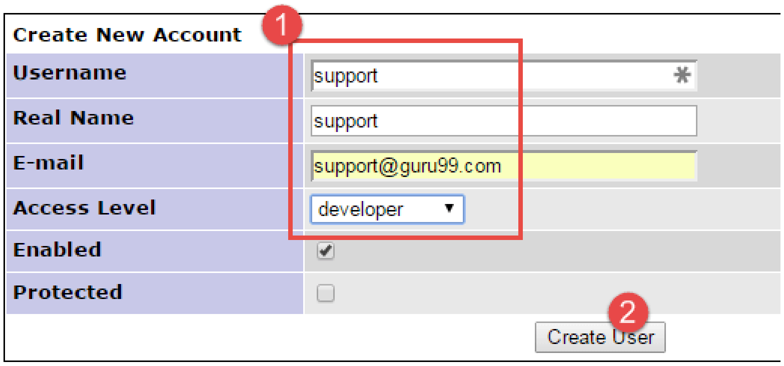


Manage Accounts [1]

Username▲	Real Name	
administrator	Krishna Rungta	krishnarung

Step 2)

1. Enter
2. Username
3. Real Name
4. Email
5. Access Level
6. Click Create User

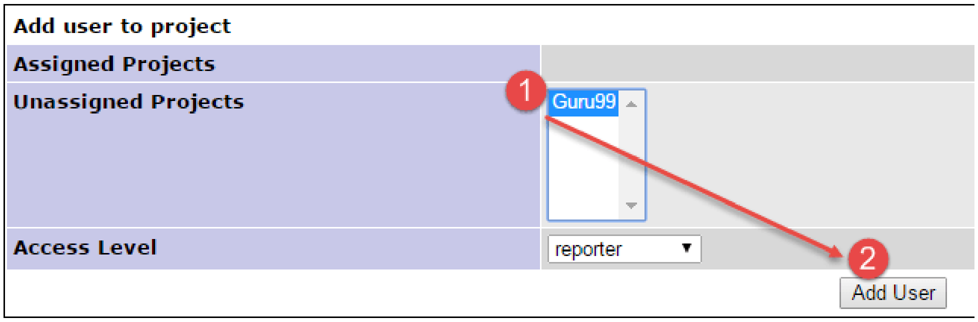


Create New Account

Username	support *
Real Name	support
E-mail	support@guru99.com
Access Level	developer ▼
Enabled	☑
Protected	□

Create User

Step 3) In the next screen assign user to the desired project.

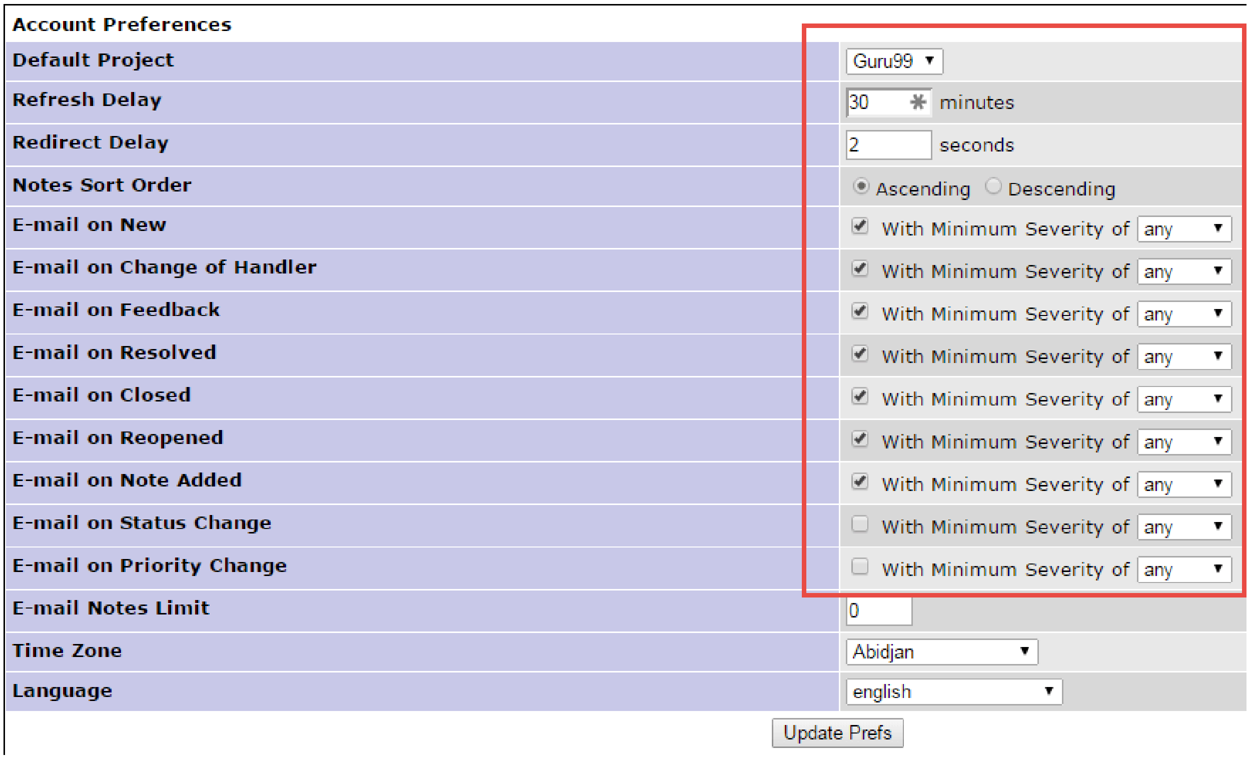


Add user to project

Assigned Projects	
Unassigned Projects	Guru99
Access Level	reporter ▼

Add User

Step 4) Update Email and other Account Preferences as desired

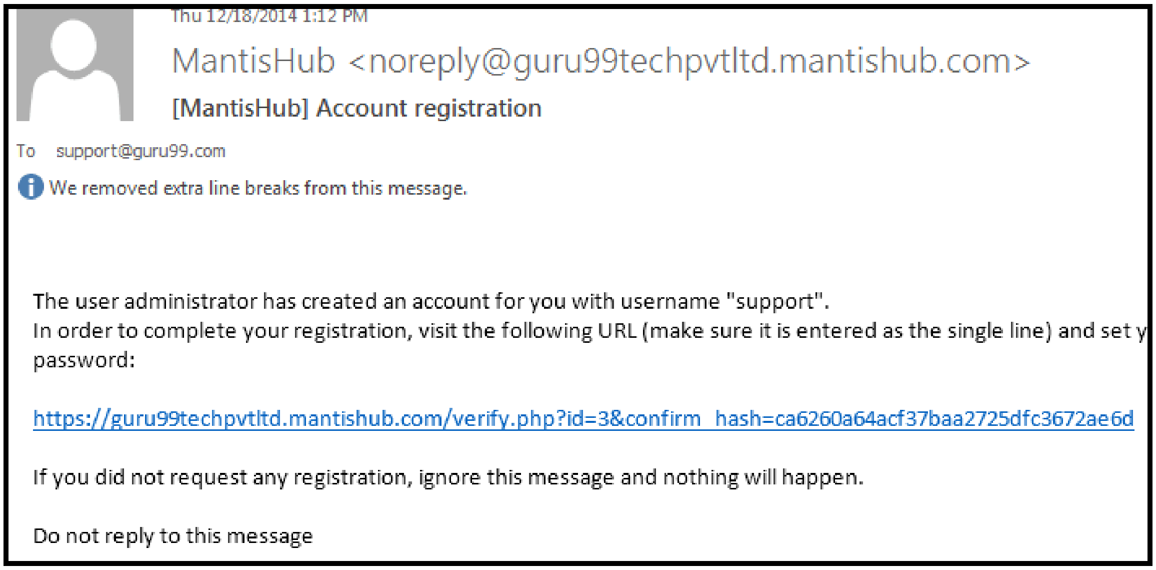


Account Preferences

Default Project	Guru99 ▼
Refresh Delay	30 * minutes
Redirect Delay	2 seconds
Notes Sort Order	● Ascending ○ Descending
E-mail on New	☑ With Minimum Severity of any ▼
E-mail on Change of Handler	☑ With Minimum Severity of any ▼
E-mail on Feedback	☑ With Minimum Severity of any ▼
E-mail on Resolved	☑ With Minimum Severity of any ▼
E-mail on Closed	☑ With Minimum Severity of any ▼
E-mail on Reopened	☑ With Minimum Severity of any ▼
E-mail on Note Added	☑ With Minimum Severity of any ▼
E-mail on Status Change	□ With Minimum Severity of any ▼
E-mail on Priority Change	□ With Minimum Severity of any ▼
E-mail Notes Limit	0
Time Zone	Abidjan ▼
Language	english ▼

Update Prefs

Step 3) Login credentials are sent to the user on him email.



Thu 12/18/2014 1:12 PM

MantisHub <noreply@guru99techpvtltd.mantishub.com>

[MantisHub] Account registration

To support@guru99.com

We removed extra line breaks from this message.

The user administrator has created an account for you with username "support".
In order to complete your registration, visit the following URL (make sure it is entered as the single line) and set y
password:

https://guru99techpvtltd.mantishub.com/verify.php?id=3&confirm_hash=ca6260a64acf37baa2725dfc3672ae6d

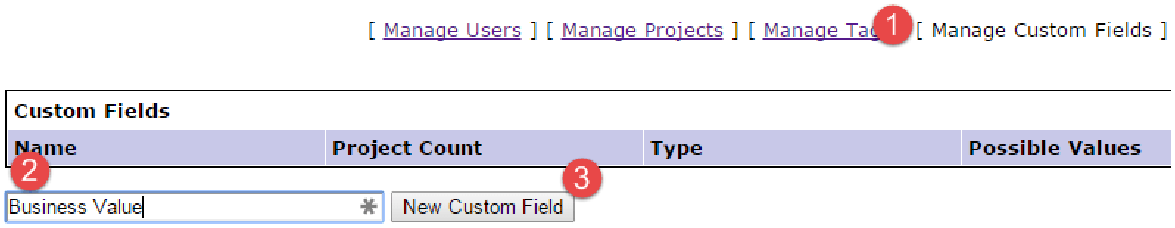
If you did not request any registration, ignore this message and nothing will happen.

Do not reply to this message

How to Create Custom Field

Step 1)

1. Go to Manage Custom Fields
2. Enter Field Name
3. Click Button "New Custom Field"



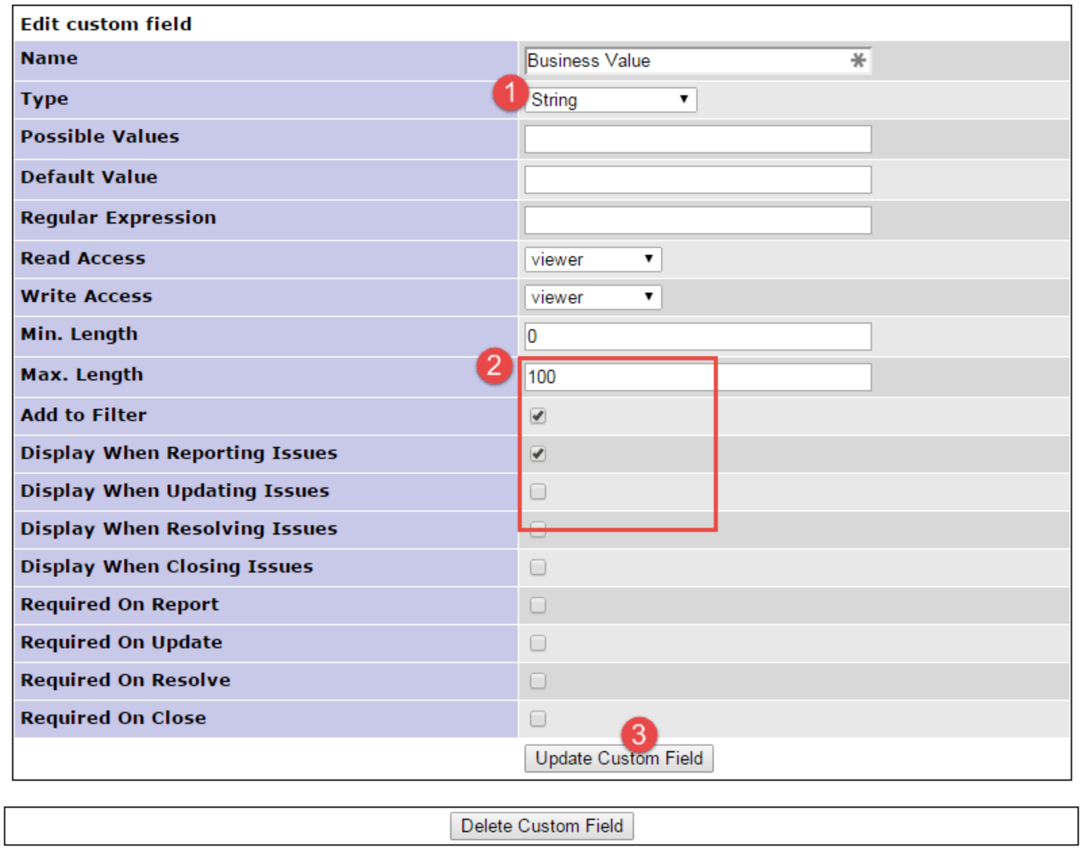
Custom Fields

Name	Project Count	Type	Possible Values

Business Value

Step 2)

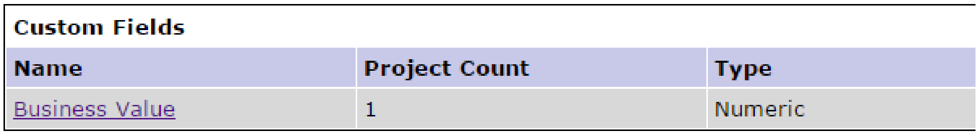
1. Select Field Type
2. Enter Field Specific Data
3. Hit Update Custom Field



Edit custom field

Name	Business Value *
Type	String ▾
Possible Values	
Default Value	
Regular Expression	
Read Access	viewer ▾
Write Access	viewer ▾
Min. Length	0
Max. Length	100
Add to Filter	☑
Display When Reporting Issues	☑
Display When Updating Issues	☐
Display When Resolving Issues	☐
Display When Closing Issues	☐
Required On Report	☐
Required On Update	☐
Required On Resolve	☐
Required On Close	☐

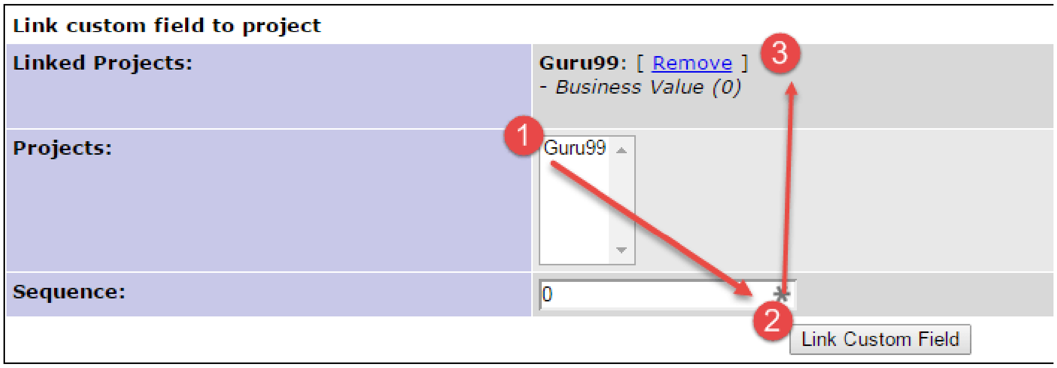
Step 3) Custom Field is created



Custom Fields

Name	Project Count	Type
Business Value	1	Numeric

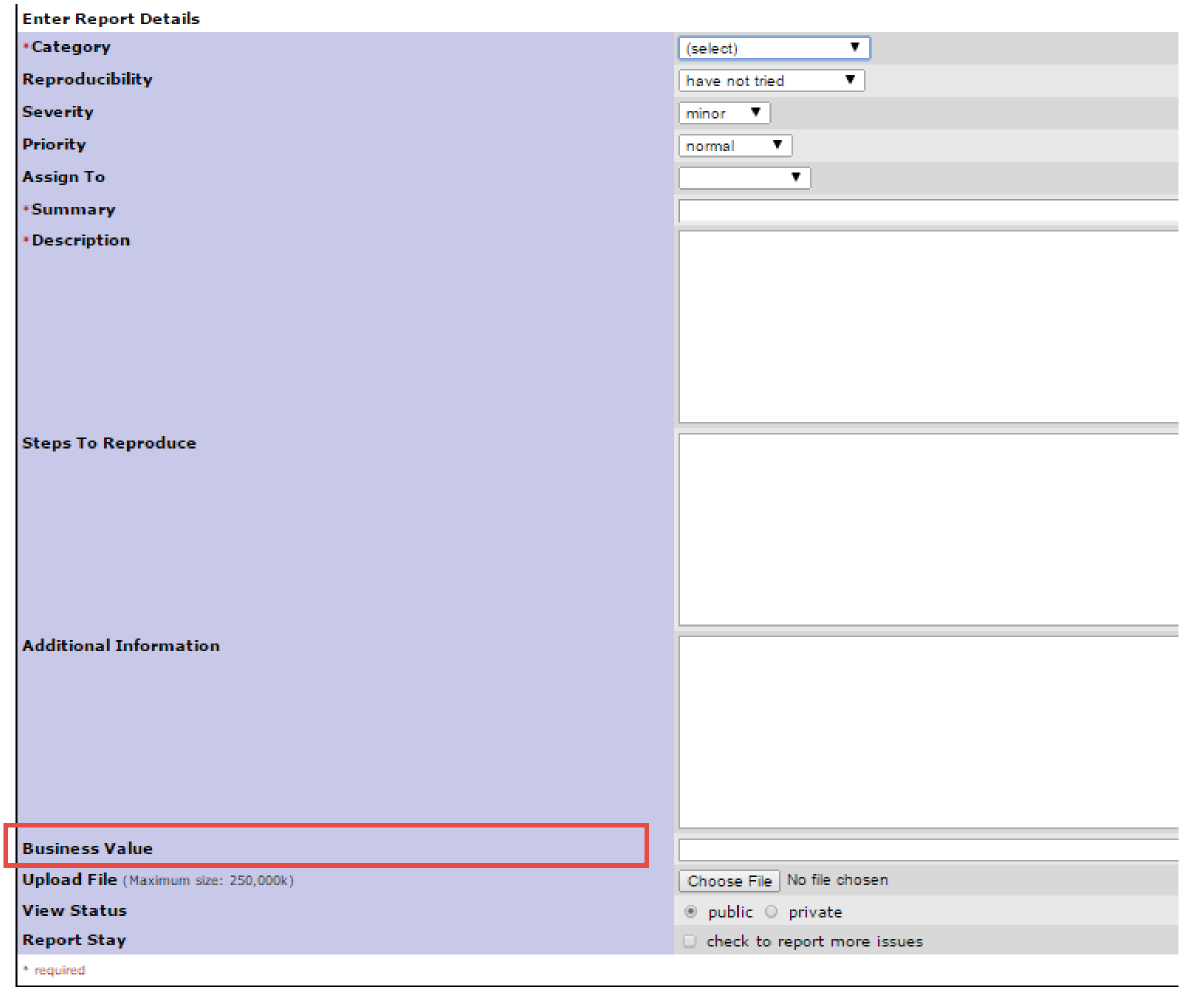
Step 4) Click the Custom Field Again and Link Custom Field to your Project



Link custom field to project

Linked Projects:	Guru99: [Remove] - *Business Value (0)*
Projects:	Guru99
Sequence:	0 *

Step 5) In the Report Issue section, the new custom field is reflected



Enter Report Details

*Category	(select) ▼
Reproducibility	have not tried ▼
Severity	minor ▼
Priority	normal ▼
Assign To	▼
*Summary	
*Description	
Steps To Reproduce	
Additional Information	
Business Value	
Upload File (Maximum size: 250,000k)	Choose File No file chosen
View Status	● public ○ private
Report Stay	□ check to report more issues

* required

Mantis Interview Question

1. **Question 1. What Is Mantis Bug Tracker?**

   Answer :

   MANTIS is a free, open source bug tracking software. It can be used to track bug for various software projects. You can easily download and install the Mantis for your use. Mantis now also provides a hosted version of the software.

2. **Question 2. Mention The Salient Features Of The Mantis Bug Tracker?**

   Answer :

   Some salient features of Mantis are:

   Email notifications: It sends out emails of updates, comments, resolutions to the concerned stake holders.

   Access Control: You can control user access at project level

Customize: You can easily customize Mantis as per your requirements.

Mobile Support: Mantis supports iPhone, Android and Windows Phone Platforms.

Plugins: An ever-expanding library of plugins to add custom functionality to Mantis.

3. **Question 3. What Is The Difference Between New And Open Bug?**

Answer :

New: In finding an issue in testing, all relevant checks are performed, like whether it was occurring in the previous release. If the issue is so, reopening the bug and creating new defect is done by changing the status of that defect to ‘new’.

Open: When a new bug is opened, development or project manager revises it, followed by assigning the task to the developer, and changes the status from ‘new’ to ‘open’.

When the bug is found by the tester, the status is marked as NEW. Once this bug is verified by the test lead for its validity, the status is changed to OPEN.

4. **Question 4. Explain The Different Types Of Bugs?**

Answer :

The different bugs are:

Show-stopper / critical bugs: The core dumps, products abnormally shuts down and no work around will be found out, like OS automatic freezing.

Major Bugs: The work around is found, but the implementation cant be done, like performance degradency.

Medium Bugs: These bugs include database errors, link errors, low response time

Low/minor Bugs: These bugs are typos, simple GUI errors.

5. **Question 5. Mention Some Tips For Bug Tracking?**

Answer :

- If any bug is confusing, it should be discussed with the developer.
- The bugs should follow a proper cycle until closed.
- If any bug closed by developer without fixing, it should be reopened.

- Never assume anything while tracking a bug.

6. **Question 6. How To Write Good Bug Tracking Report?**

Answer :

Title should be clear: A good title is a must, which the essence of the bug report be well grasped by the developer.

One Report per Bug : A bug report should have only one bug and not more or no less. In case more bugs are placed, some of the bugs may be overlooked.

Minimum steps to reproduce the Bug: This is a very important tip. Developers need to get the problem in the shortest possible time. So the tester needs to help them for doing just that task. Testers need to have few rounds of testing and be able to produce the problems using minimum steps.

Expected and observed results: A bug report should always contain the expected and the observed result. Testers should take the responsibility to explain the developers that what went wrong.

The build that the problem occurs: It is common in the daily builds. If the exact problematic build is not specified by the tester, it is hard for developers to resolve an already-solved problem.

Pictures: ‘A picture is worth a thousand words’! it is better to have a clear picture that perfectly illustrates the problem.

7. **Question 7. What Is The Difference Between Bug And Defect?**

Answer :

- A bug is getting a problem at the time of testing, where as a defect is problem that got by the customer in production time
- A bug is a fault in a program which causes it to behave abruptly. Bugs are usually found either during unit testing done by developer of module testing by testers.
- A defect is found when the application does not conform to the requirement specification. A defect can also be found when the client or user is testing.

8. **Question 8. What Is The Process Of Bug Cycle?**

Answer :

The following steps involved in the bug cycle process:

Bug investigation:when the tester find the bug

Bug assignment:when the tester assigns the bug to the developer

Bug fixing: when the bug is fixed by the developer

Bug retesting:when the developer fixes the bug and given to the tester

Bug closing and reopening:when the tester retested and the status is changed to 'closed'.

9. **Question 9. What Is The Diff Status Of Bug Cycle?**

Answer :

- A bug when found by the tester is marked as ***NEW***.
- This bug is checked for its validity by the test lead to mark it as ***OPEN***.
- The bug is then ***ASSIGNED*** to the developer.
- The developer on fixing the bug gives it to the tester, & it is ***REOPENed*** for regression testing.
- After successful testing if bug not found then tester ***CLOSE*** the bug.

10. **Question 10. How To View Specific Projects In Mantis Bug Tracker?**

Answer :

- Log in with your username and password. If you've forgotten your credentials, you can reset your password using your email address from the link on the login page.
- To see your project's main page, you must select its name in the drop-down menu displayed in the upper right corner of the screen. Once the page loads, click on the tab 'See issues' to see all the previous entry bugs for this project.

11. **Question 11. Explain How To Report A Bug In Mantis Bug Tracker?**

Answer :

1) To report a new issue, click on ***'Report Issue'*** in the main menu bar (shown below.)

2) A form will open titled "***Enter Report Details***". You must fill in every field that it contains. The following steps describe how to do this

3) First, you must always select ***'All projects – general'*** in the *Category drop-down menu.

4) Select the ***reproducibility*** from the second dropdown. Keep in mind that as much as you can, you should find the bug's circumstances before logging it. Ideally, you should be able to reproduce it 100%. But sometimes it is impossible. In these rare cases, you can use ***'sometimes'*** if the bug occurs randomly if you really are unable to reproduce it.

5) **Select a *severity* for the bug:**

Minor if the bug is important but not harmful to the core functioning of a feature

Major if the bug is important and compromises the user's experience

Critical if the bug completely prevents the user from continuing what he was doing. if the bug crashes site or application

6) **Select a *priority* for the bug:**

'Low' if the bug has very little impact on the user's experience

'Normal' if it needs fixing soon but it is not urgent

'High' if it needs fixing relatively soon

'Urgent' if it is a top priority and should be quickly

'Immediate' if it needs fixing right now

7) Fill in the 3 forms under "***Select Profile***". These are:

Select a Platform: the device you used for your tests

Select an Operating System: the operating system of the device

Select an OS version: the version number of the operating system

8) Leave the ***'Assign to'*** drop-down blank unless told otherwise.

9) In the summary field, enter a ***title*** for your bug. The title should begin with the name of the section of the app/website/game in which the bug happens between brackets [], followed by a brief description of the bug.

10) Update the description field the detailed info about the bug.

11) The ***'Steps to reproduce'*** section should contain the steps that a developer or other tester should follow to reproduce the issue

12) The ***'Additional information'*** field must be used to inform the developer of what device/browser and on what device/browser version the bug was found

13) Use ***'Upload*** File' to upload your ***screenshot*** or video

14) View status should be left ***public*** and the 'report stay' checkbox should be left unchecked unless you have another issue to enter immediately.

12. **Question 12. Does Mantis Provide An Xml-rpc Interface?**

Answer :

No, but it provides a SOAP webservice interface.

13. **Question 13. Is There A Mantis Version For Smart Phones / Pdas?**

Answer :

Yes, MantisWAP provides a very light weight interface for Mantis access through a smart phone or PDA. It is optimize for speed and minimization of consumed bandwidth.

14. **Question 14. How To Add A Mantis Bug Tracker?**

Answer :

Under Manage Tab, go to Manage Projects:

- Enter Project Name
- Select Status
- Enter Description
- Click Add Project
-

15. **Question 15. How To Create User In Mantis Bug Tracker?**

Answer :

Go to Manage > Manage Users and Click on 'Create New Account' and update below fields:

- Username
- Real Name
- Email
- Access Level
- Click Create User

In the next screen assign user to the desired project.Update Email and other Account Preferences as desired. After that the login credentials are sent to the user on their email.

16. **Question 16. How To Create Custom Field In Mantis?**

Answer :

- Go to Manage Custom Fields

- Enter Field Name
- Click Button 'New Custom Field'
- Select Field Type
- Enter Field Specific Data
- Hit Update Custom Field

Now the custom field got created. Click the Custom Field Again and Link Custom Field to your Project. After that in Report Issue section, the new custom field will be reflected.

17. **Question 17. Does Mantisbt Integrate With Source Control Tools?**

Answer :

Scmbug is a system that integrates software configuration management (SCM) with bug-tracking. It aims to be a universal tool that will glue any source code version control system (such as CVS, Subversion, and Arch) with any bug-tracking system (such as Bugzilla, MantisBT, and Request Tracker).

18. **Question 18. How Do We Get Help For Mantis?**

Answer :

- Use MantisBT Forums or mantisbt-help at lists dot sourceforge dot net for posting questions. Our preference goes to the forums.
- Use mantisbt-dev at lists dot sourceforge dot net for development related discussions.
- Use Mantis Bug Tracker for reporting feature requests and bugs.
- Use MantisBT Manual comments to identify a problem with the documentation or to contribute hints that are ought to be part of the documentation.
- Use mantisbt-lang at lists dot sourceforge dot net for discussions relating to localisation to a specific language.

19. **Question 19. How Do We Contribute Patches?**

Answer :

- If you are about to implement a major feature, it is strongly recommended to get in touch with the developers. Such interaction will make it much more likely that your mods be applied to MantisBT distribution.
- Patches should be attached to a Mantis Bug Tracker issue.
- A patch should be attached as a single zip or .tar.gz file.
- A patch should include a copy of all new files / modified files as well as a diff (patch) file (if possible).

- A patch should only contain one feature. If a patch has a mixed bag of features, it makes it harder to understand and apply.
- If you require feedback from developers relating to the patch, then post a message to mantisbt-dev at lists dot sourceforge dot net which refers to the issue including your patch. Ideally, the feedback will be added to the issue as notes. If not, then the patch owner should summarise the feedback as notes.
- Fixes and updates to the patch should be attached to the same issue as should be marked clearly. With every attachment there should be a note that refers to the file name and specifies the updates done.
- Join MantisBT IRC channel for realtime discussions with developers.
- Patches should following MantisBT coding style.
- Keep your patches up-to-date as new MantisBT versions are released.

20. **Question 20. How Do We Contribute Localisation To A New Language?**

Answer :

- Use strings_english.txt as the basis for the new localisation file.
- Consider using UTF8 encoding.
- Submit an issue into the Mantis Bug Tracker and attach a compressed version of the file.

The process of sending Code from eclipse to GIT local repository daily is called as ***Commit***.
And The process of sending Code from GIT local repository to remote GIT HUB repository is called as ***Push***.(check out)

The process of extracting Code from remote GIT HUB repository to GIT local repository to is called as ***Pull***.(check in)
The process of extracting Code from GIT local repository to eclipse to daily is called as ***Request*** .

GIT is provided as Plug-in for eclipse.

GIT – local Repository

GIT Hub – Remote Repository

Project Name and Repository Name is same

Need to install GIT plug in eclipse thro market place → Search GIT → and install EGIT integration with eclipse

Step 1*. *Create/ clone GIT repository in eclipse

search option (Quick Access → GIT Repositories)→ Clone a GIT repository → provide URL

Step 2. Commit and Push your Project to GIT & GIT Hub

(*Only 1st Time*) Select your project → right click → team → share project →select GIT repository from drop down → finish

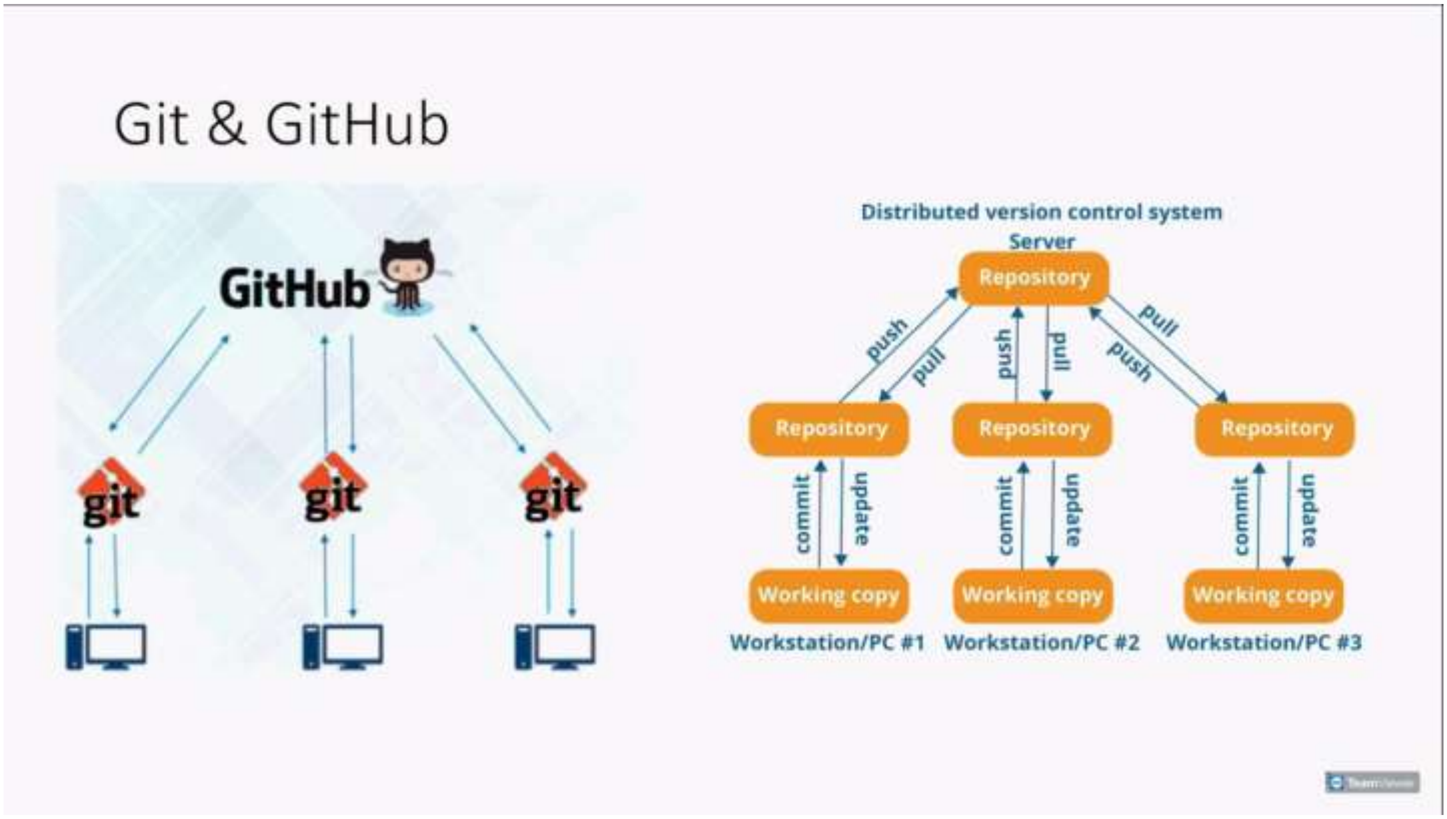
(from 2nd time and every time whenever to do changes) Select your project → right click → team → commit (*wen u modifies the code)*

Step 3. Pushing your code from GIT to GIT Hub Repository

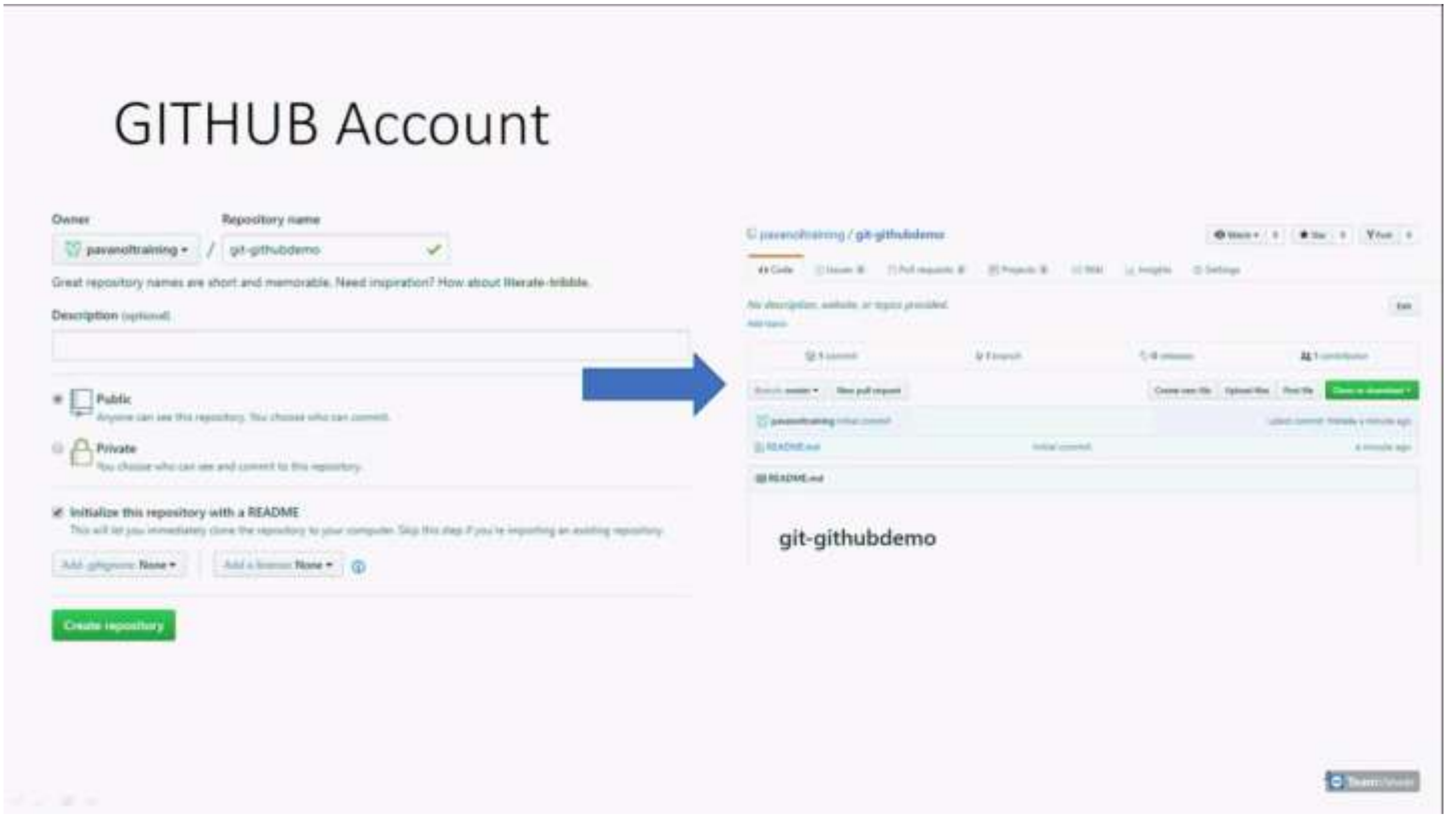
Select your project → right click → team → commit ---- select your all files from unstaged changes box – drag n drop them to changed stages → click on commit and push

Step 4. Pulling files from GitHub to Git Reopsitory

Select your project → right click → team →pull



Git & GitHub



GITHUB Account

Owner: pavanoltraining / Repository name: git-githubdemo ✓

Great repository names are short and memorable. Need inspiration? How about literate-tribble.

Description (optional)

● Public
Anyone can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

☑ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾ | Add a license: None ▾

pavanoltraining / git-githubdemo

No description, website, or topics provided.

Add topics

1 commit | 1 branch | 0 releases | 1 contributor

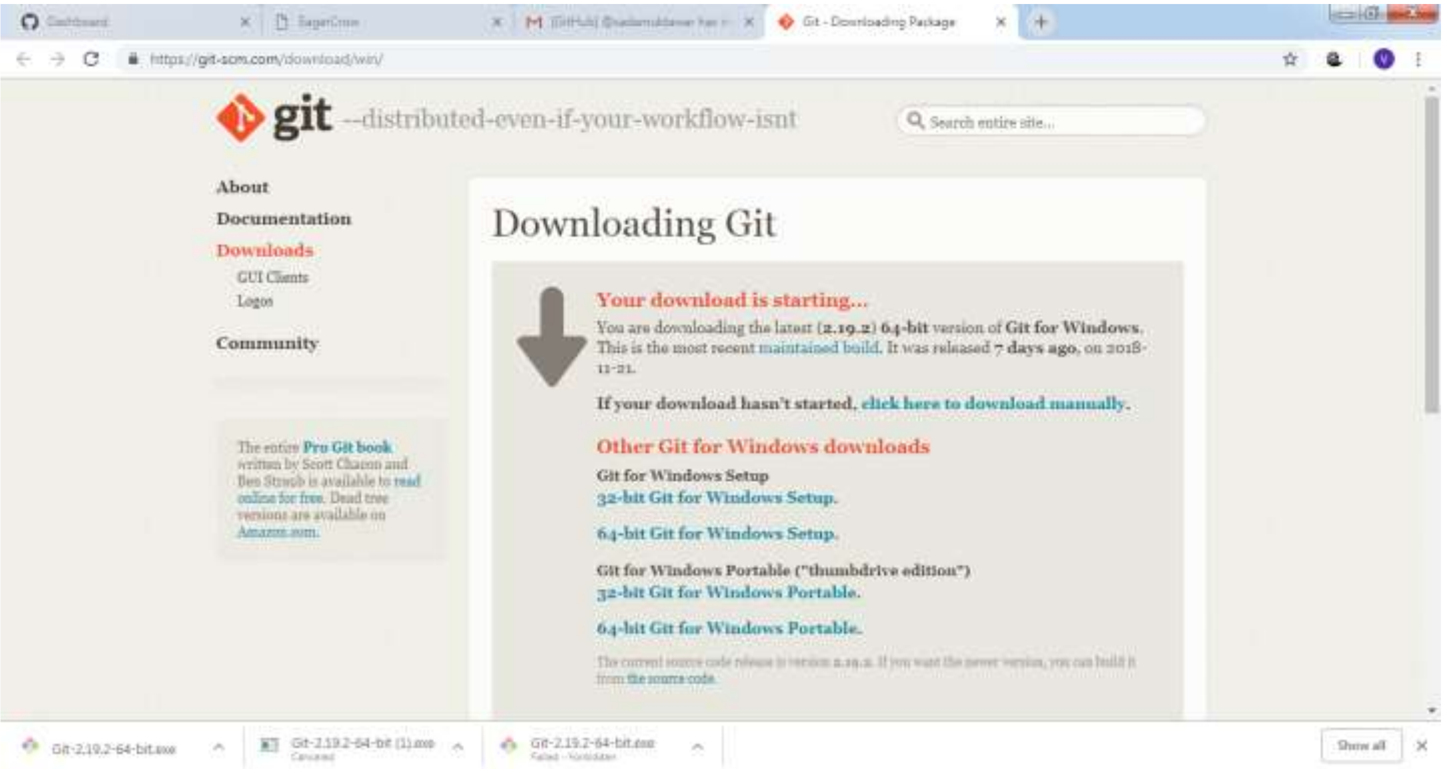
pavanoltraining Initial commit — Latest commit ... a minute ago

README.md — Initial commit — a minute ago

README.md

git-githubdemo

GIT Installation, download GIT from following link
http://git-scm.com/download/win/



Git - Downloading Package
https://git-scm.com/download/win/
git --distributed-even-if-your-workflow-isnt
Search entire site...
About
Documentation
Downloads
GUI Clients
Logos
Community
Git-2.19.2-64-bit.exe
Git-2.19.2-64-bit (1).exe
Git-2.19.2-64-bit.exe
Show all

Downloading Git

Your download is starting...

You are downloading the latest (**2.19.2**) **64-bit** version of **Git for Windows**. This is the most recent maintained build. It was released **7 days ago**, on 2018-11-21.

If your download hasn't started, click here to download manually.

Other Git for Windows downloads

Git for Windows Setup

32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

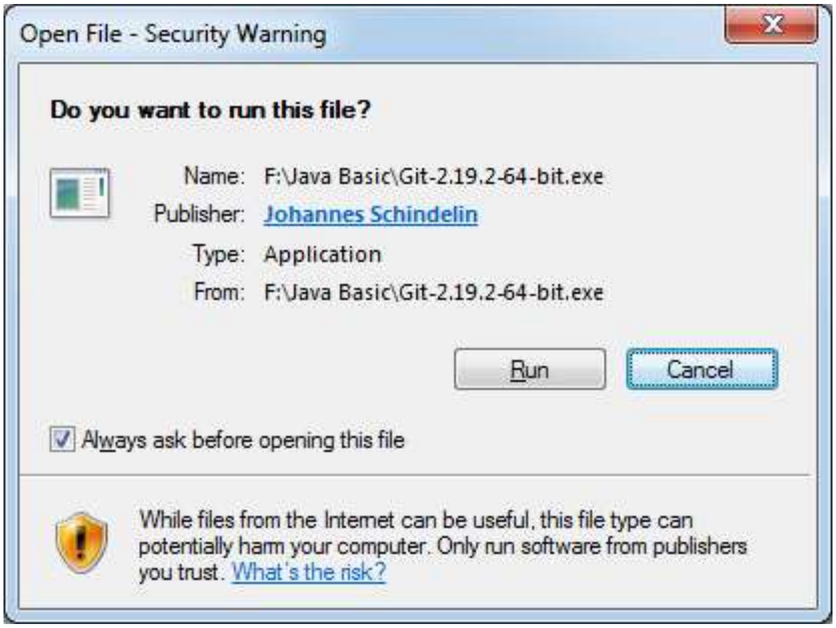
Git for Windows Portable ("thumbdrive edition")

32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

The current source code release is version 2.19.2. If you want the newer version, you can build it from the source code.

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.



Open File - Security Warning

Do you want to run this file?

Name: F:\Java Basic\Git-2.19.2-64-bit.exe

Publisher: Johannes Schindelin

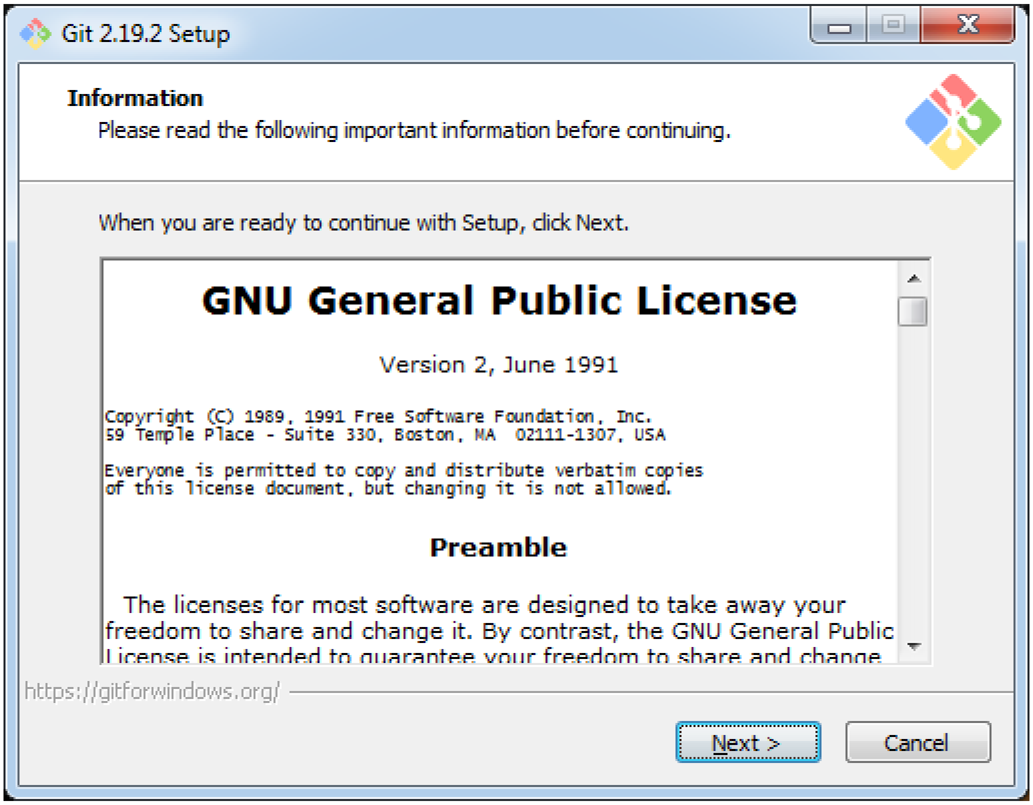
Type: Application

From: F:\Java Basic\Git-2.19.2-64-bit.exe

☑ Always ask before opening this file

While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?

Window 1.--- **Run the setup file..**



Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

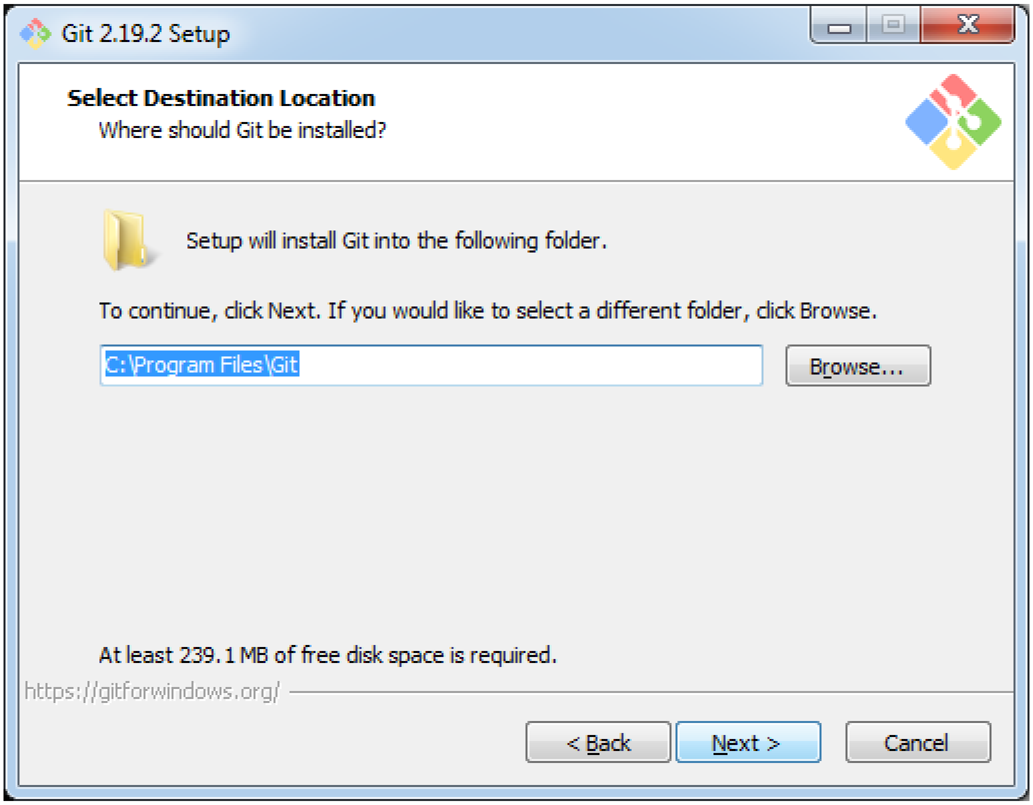
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

Window 2.--- Click on **Next** button



Select Destination Location

Where should Git be installed?

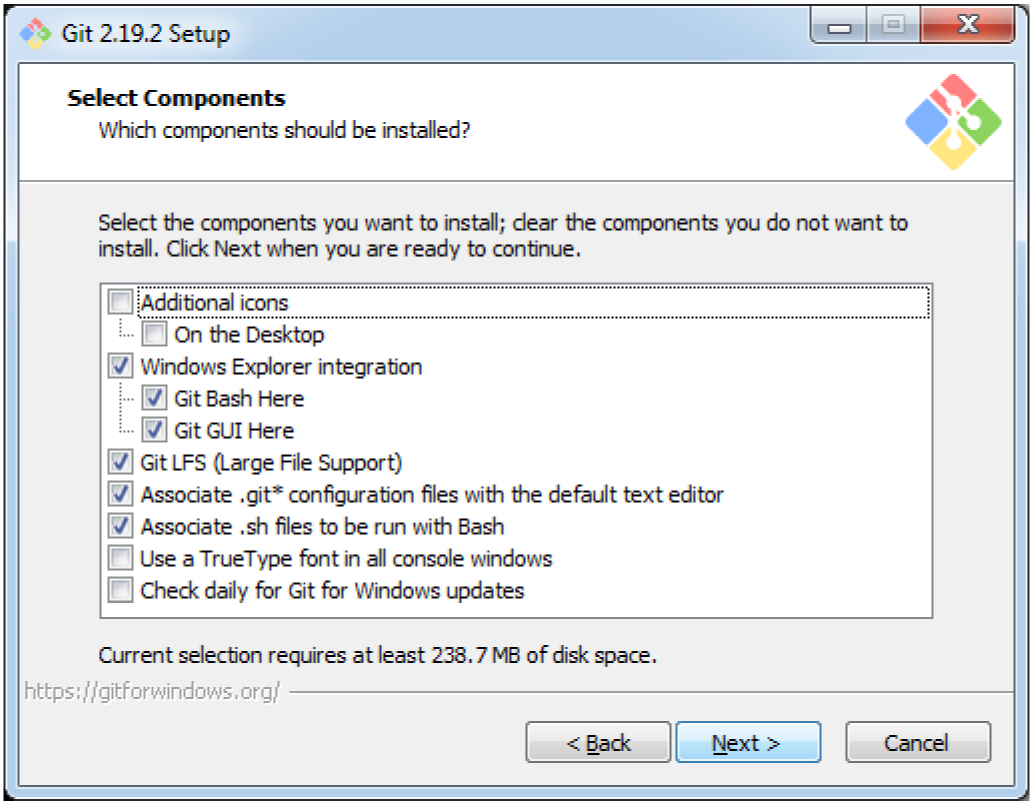
Setup will install Git into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Git

At least 239.1 MB of free disk space is required.

Window 3. -- Click on **Next** button



Git 2.19.2 Setup

Select Components

Which components should be installed?

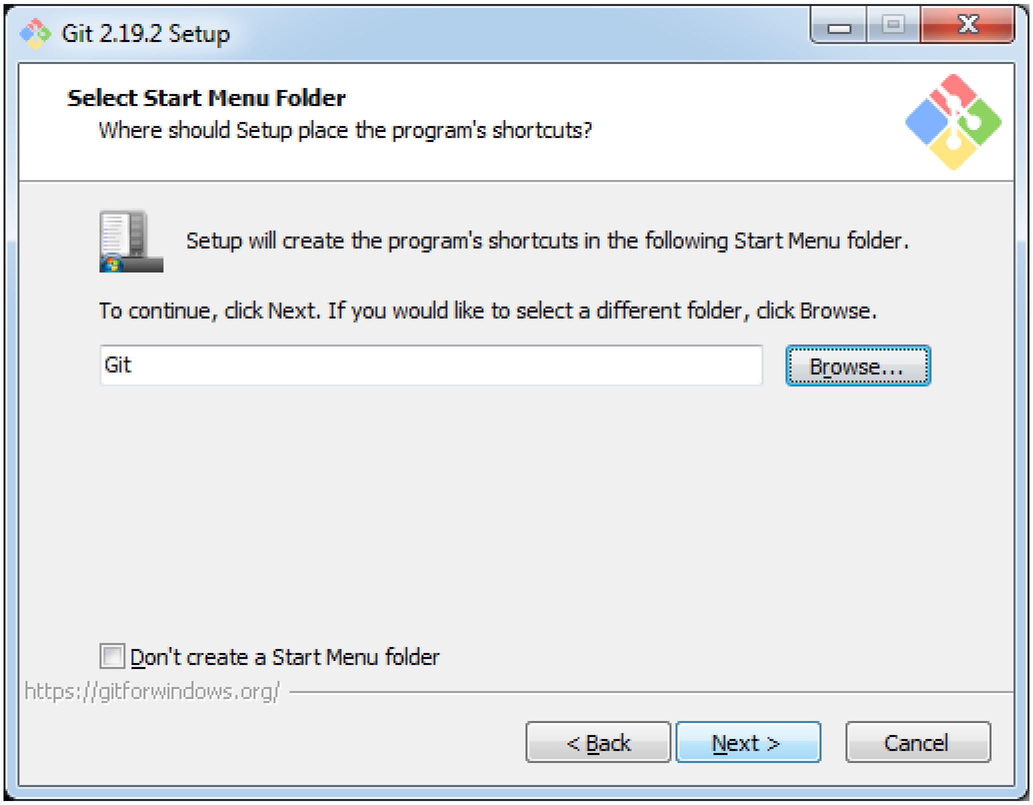
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ☐ Additional icons
  - ☐ On the Desktop
- ☑ Windows Explorer integration
  - ☑ Git Bash Here
  - ☑ Git GUI Here
- ☑ Git LFS (Large File Support)
- ☑ Associate .git* configuration files with the default text editor
- ☑ Associate .sh files to be run with Bash
- ☐ Use a TrueType font in all console windows
- ☐ Check daily for Git for Windows updates

Current selection requires at least 238.7 MB of disk space.

https://gitforwindows.org/

Window 4. -- **Check** the box for **Git Bash Here** and **Git GUI Here** and click **Next…**



Git 2.19.2 Setup

Select Start Menu Folder

Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

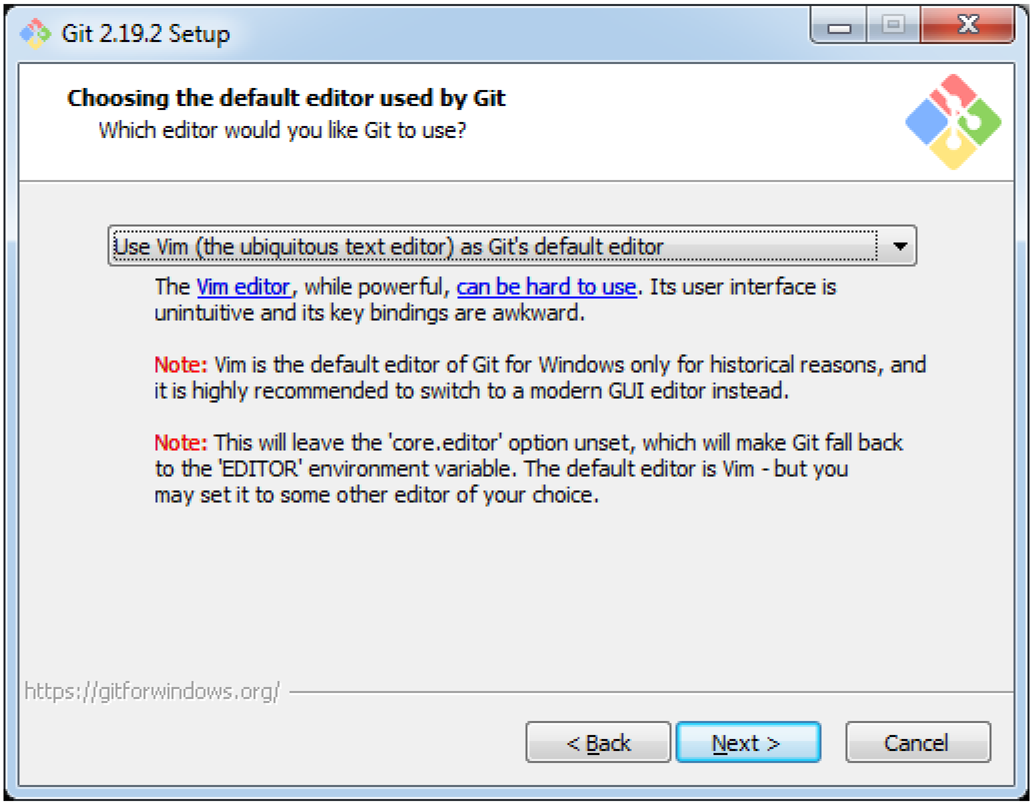
To continue, click Next. If you would like to select a different folder, click Browse.

Git

☐ Don't create a Start Menu folder

https://gitforwindows.org/

Window 5.--- Click on **Next** >



Choosing the default editor used by Git

Which editor would you like Git to use?

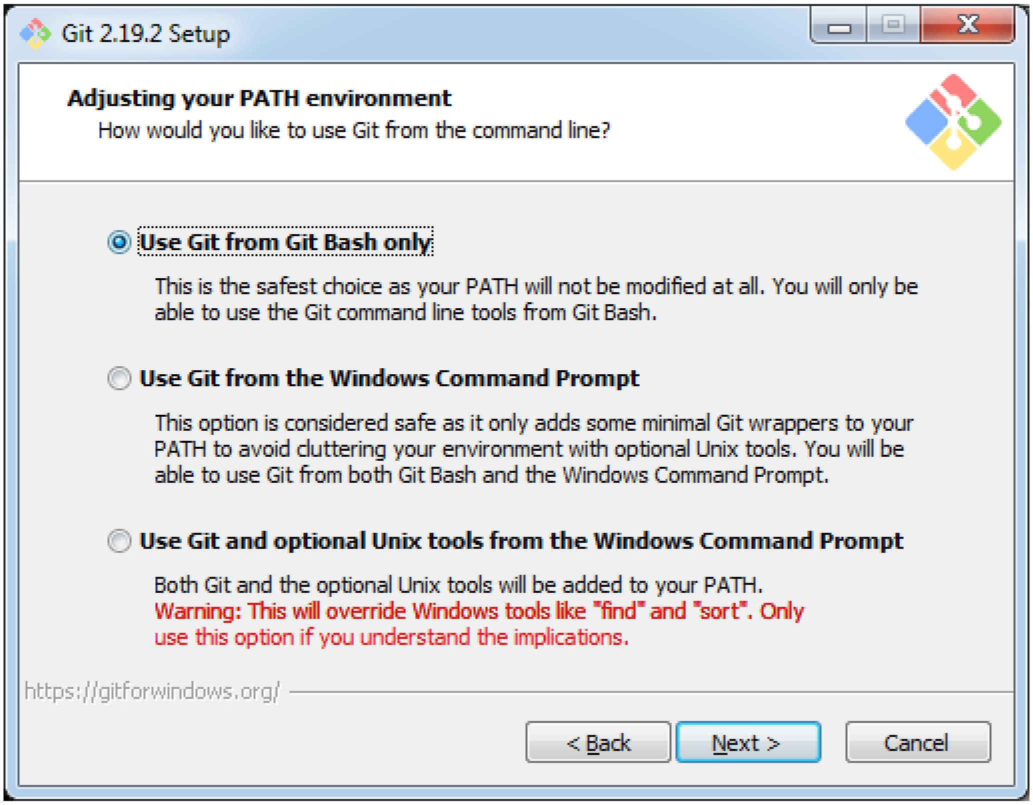
Use Vim (the ubiquitous text editor) as Git's default editor

The Vim editor, while powerful, can be hard to use. Its user interface is unintuitive and its key bindings are awkward.

Note: Vim is the default editor of Git for Windows only for historical reasons, and it is highly recommended to switch to a modern GUI editor instead.

Note: This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.

Window 6. --- Click on **Next** >



Adjusting your PATH environment

How would you like to use Git from the command line?

● **Use Git from Git Bash only**

This is the safest choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

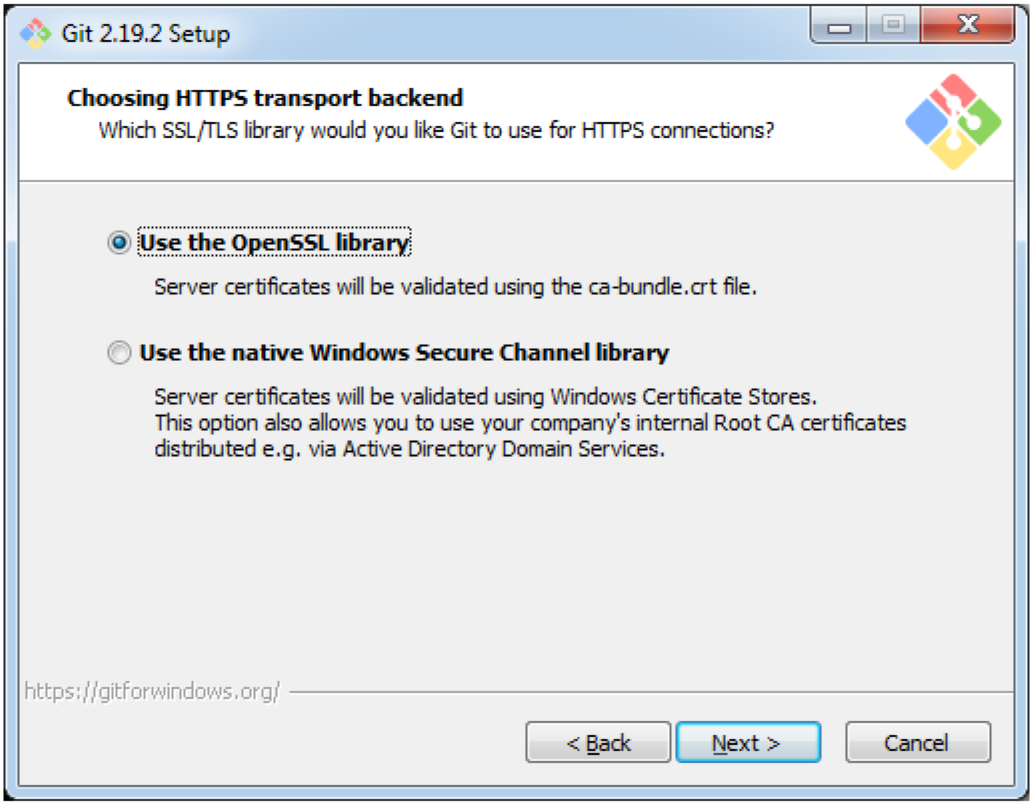
○ **Use Git from the Windows Command Prompt**

This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from both Git Bash and the Windows Command Prompt.

○ **Use Git and optional Unix tools from the Windows Command Prompt**

Both Git and the optional Unix tools will be added to your PATH. Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

Window 7.-- Select **Use GIT From Git Bash only** and click **Next** >



Choosing HTTPS transport backend

Which SSL/TLS library would you like Git to use for HTTPS connections?

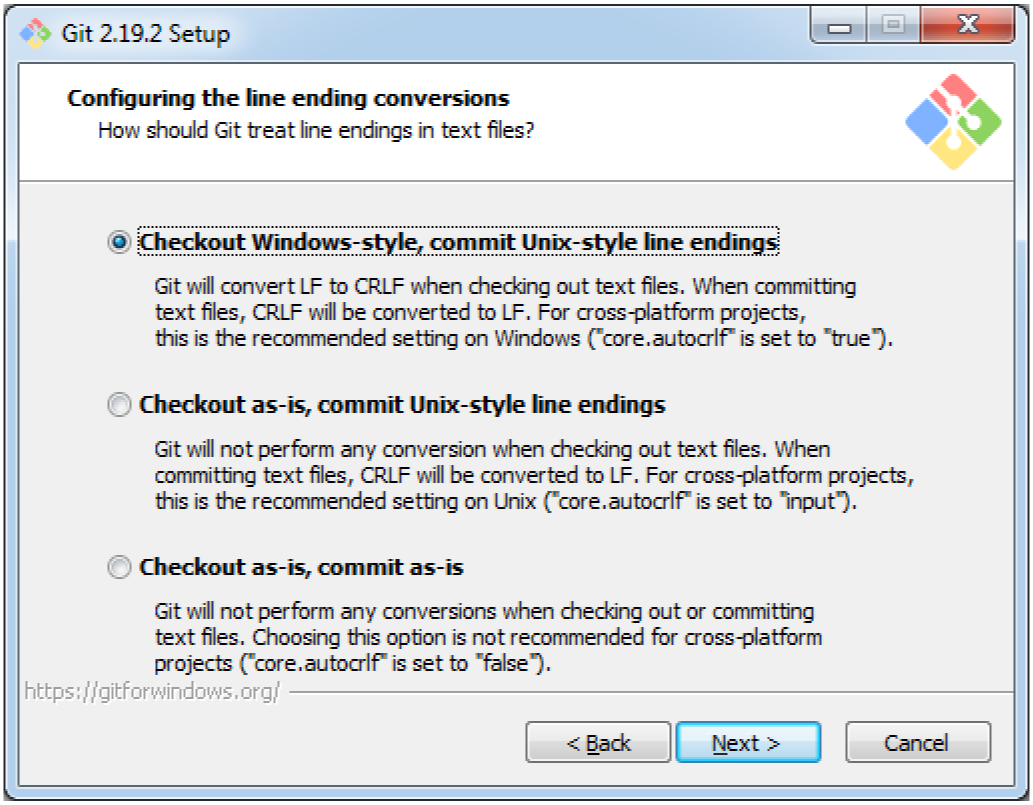
● **Use the OpenSSL library**

Server certificates will be validated using the ca-bundle.crt file.

○ **Use the native Windows Secure Channel library**

Server certificates will be validated using Windows Certificate Stores.
This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.

Window 8. -- Select **Use the OpenSSL library** and click **Next >**



Configuring the line ending conversions

How should Git treat line endings in text files?

● **Checkout Windows-style, commit Unix-style line endings**

Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

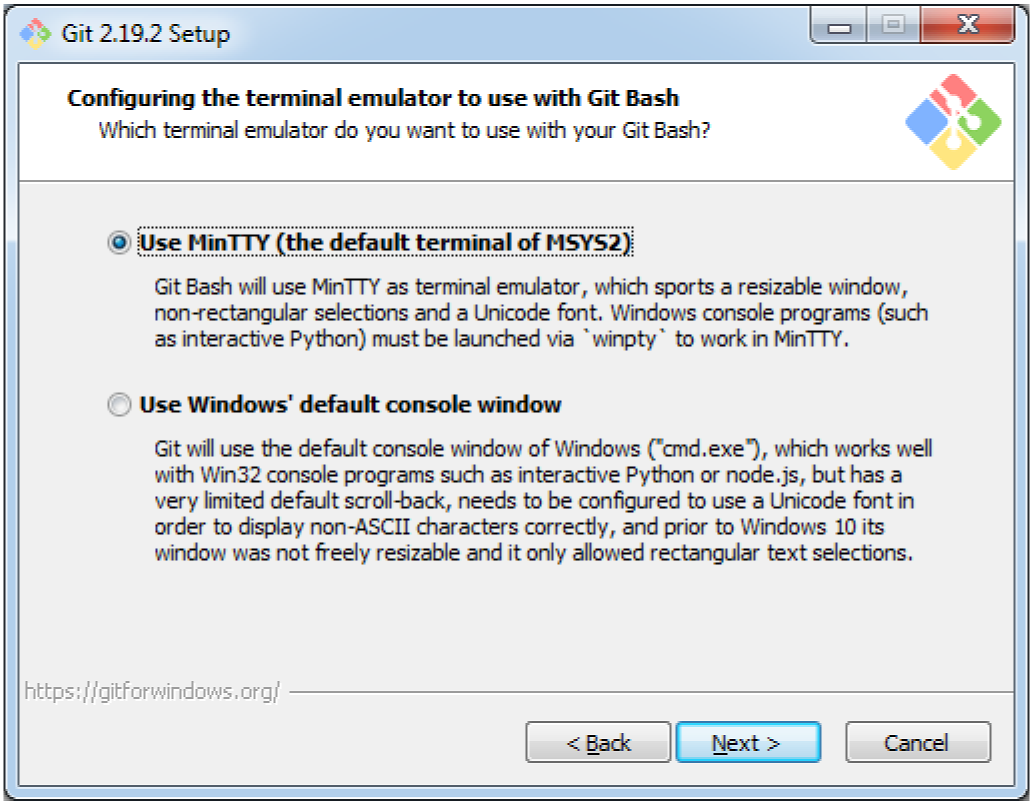
○ **Checkout as-is, commit Unix-style line endings**

Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

○ **Checkout as-is, commit as-is**

Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

Window 9. → Select **Checkout Windows-Style, commit Unix-style line endings** and click **Next >**



Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?

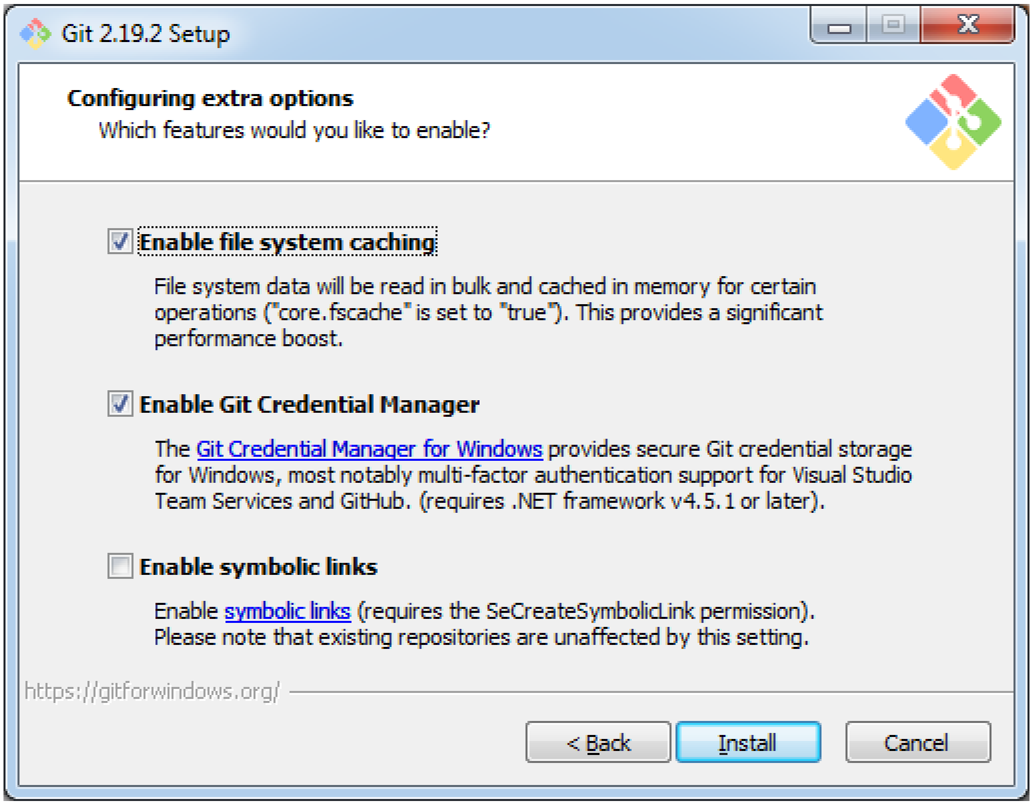
● **Use MinTTY (the default terminal of MSYS2)**

Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

○ **Use Windows' default console window**

Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.

Window 10. → select **Use MinTTY (the default terminal of MSYS2)** and click **Next>**



Configuring extra options
Which features would you like to enable?

☑ **Enable file system caching**

File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.

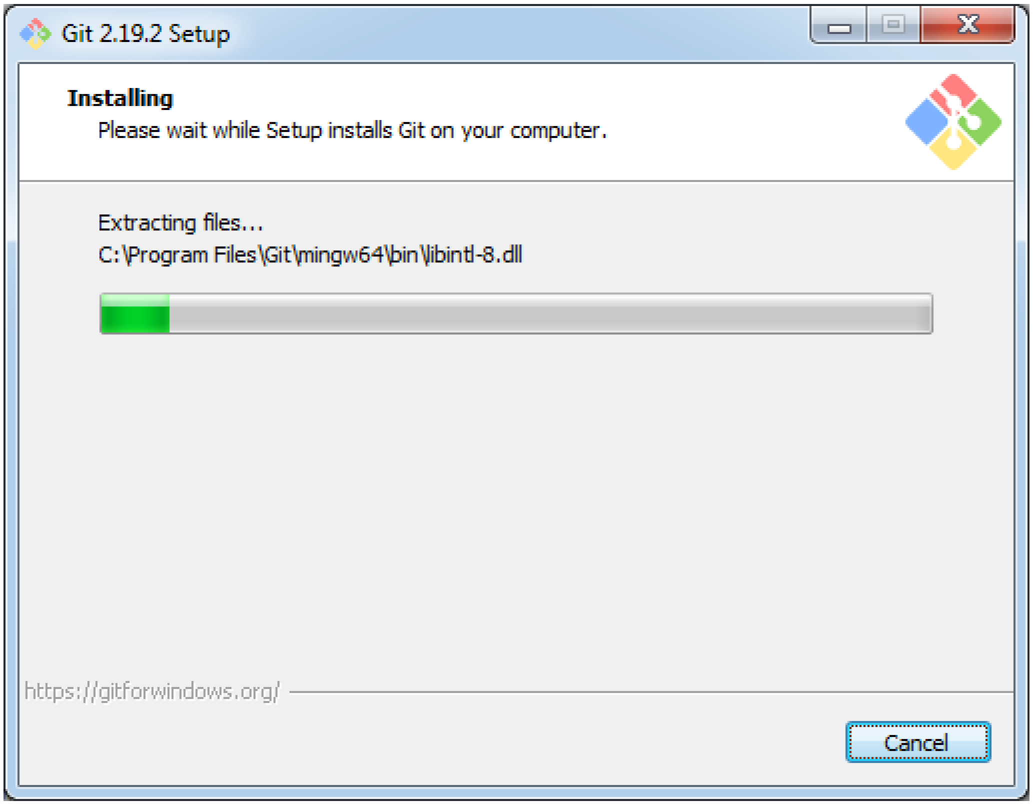
☑ **Enable Git Credential Manager**

The Git Credential Manager for Windows provides secure Git credential storage for Windows, most notably multi-factor authentication support for Visual Studio Team Services and GitHub. (requires .NET framework v4.5.1 or later).

□ **Enable symbolic links**

Enable symbolic links (requires the SeCreateSymbolicLink permission). Please note that existing repositories are unaffected by this setting.

Window 11 → **Select Enable file system caching** and **Enable Git Credential Manager** click **Install**



Git 2.19.2 Setup

Installing

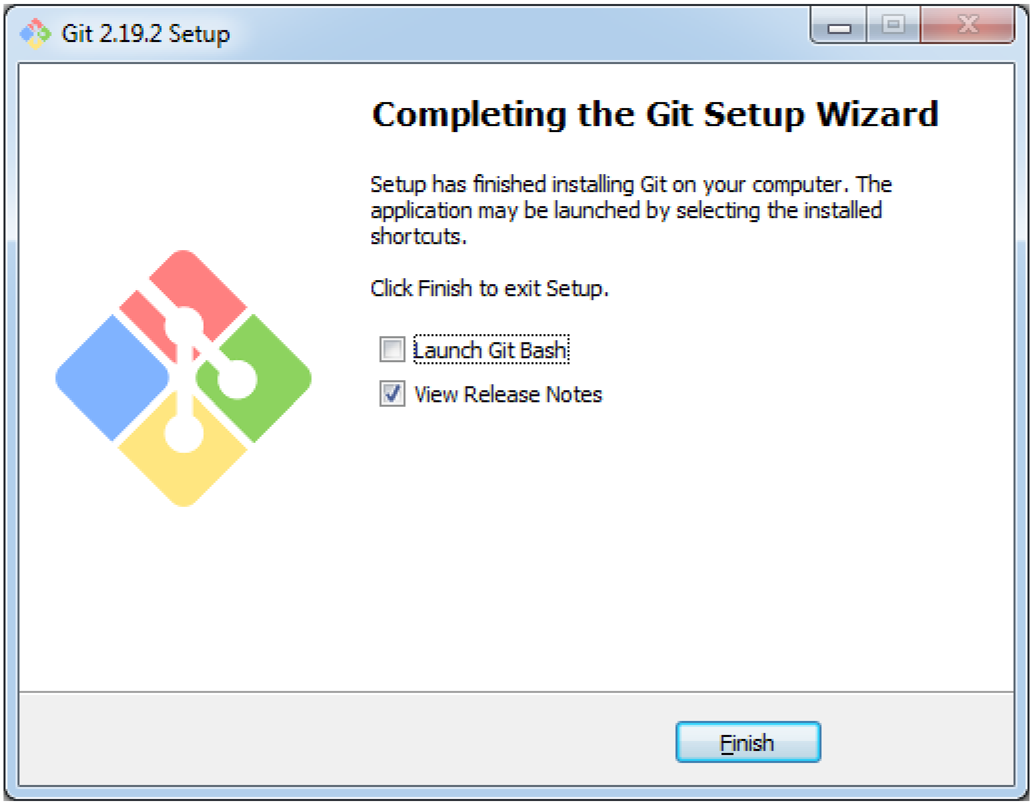
Please wait while Setup installs Git on your computer.

Extracting files...

C:\Program Files\Git\mingw64\bin\libintl-8.dll

https://gitforwindows.org/

Window 12. → installing window



Git 2.19.2 Setup

Completing the Git Setup Wizard

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

□ Launch Git Bash

☑ View Release Notes

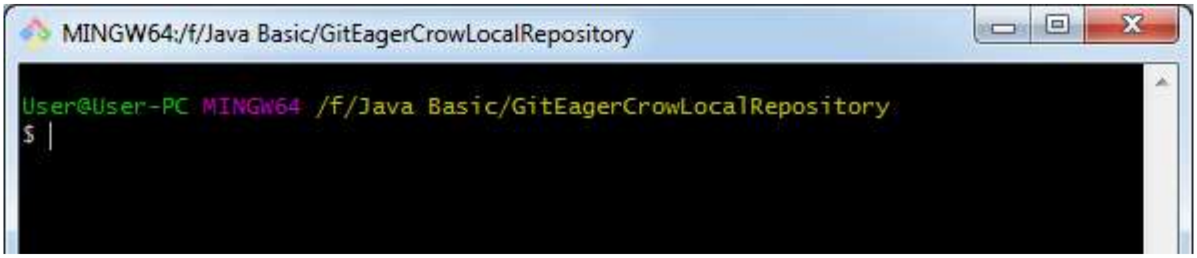
Window 13. → click on **Finish**

Now follow following steps to create GitHub Repository and Git Local Repository

1. Sign up on **https://github.com/**
2. Create your account and create new repository by clicking on **Start Project**
3. After creation of repository click on **Clone or Download** and copy **url**
4. e.g. https://github.com/vaibhav01M/EagerCrow.git ← **GitHub** Repository

Create a Folder on your Computer as a Git Local Repository and name it as per your convenience

1. e.g. **GitEagerCrowLocalRepository** enter in it
2. after entering local repository right click and click on **Git Bash Here.** Window like command prompt will appear as follows



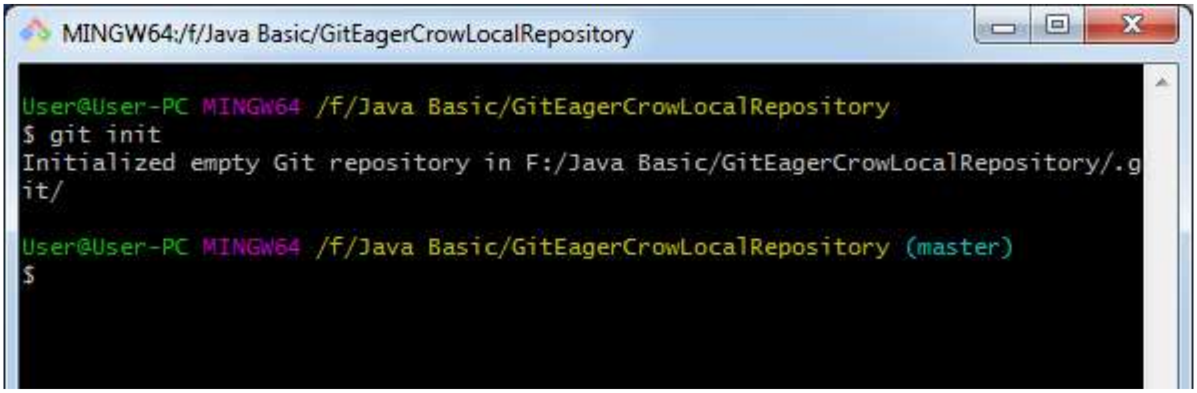
MINGW64:/f/Java Basic/GitEagerCrowLocalRepository

```
User@User-PC MINGW64 /f/Java Basic/GitEagerCrowLocalRepository
$ |
```

3. Initialize **GitEagerCrowLocalRepository** (directory) repository as local Git repository by using following command.

   Command : $ git init press enter…

4. As soon as you execute above command, empty folder/Directory will become Git Local repository. See following window.



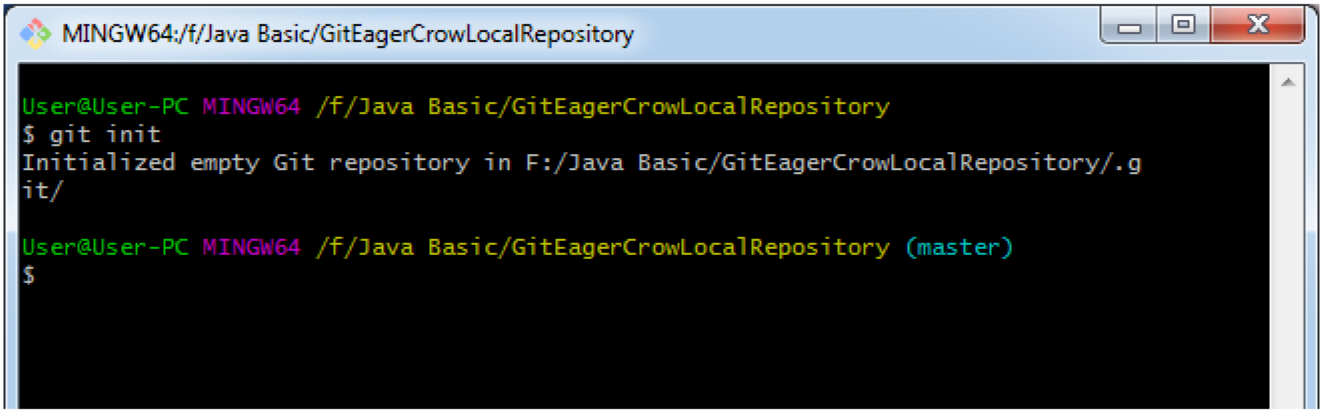
MINGW64:/f/Java Basic/GitEagerCrowLocalRepository

```
User@User-PC MINGW64 /f/Java Basic/GitEagerCrowLocalRepository
$ git init
Initialized empty Git repository in F:/Java Basic/GitEagerCrowLocalRepository/.g
it/

User@User-PC MINGW64 /f/Java Basic/GitEagerCrowLocalRepository (master)
$
```

5. Integration of **Git** Local and **Git Hub** Remote Repository

e.g. $ git remote add origin “--------url---------”

command : $ git remote add origin “https://github.com/vaibhav01M/EagerCrow.git”
press enter

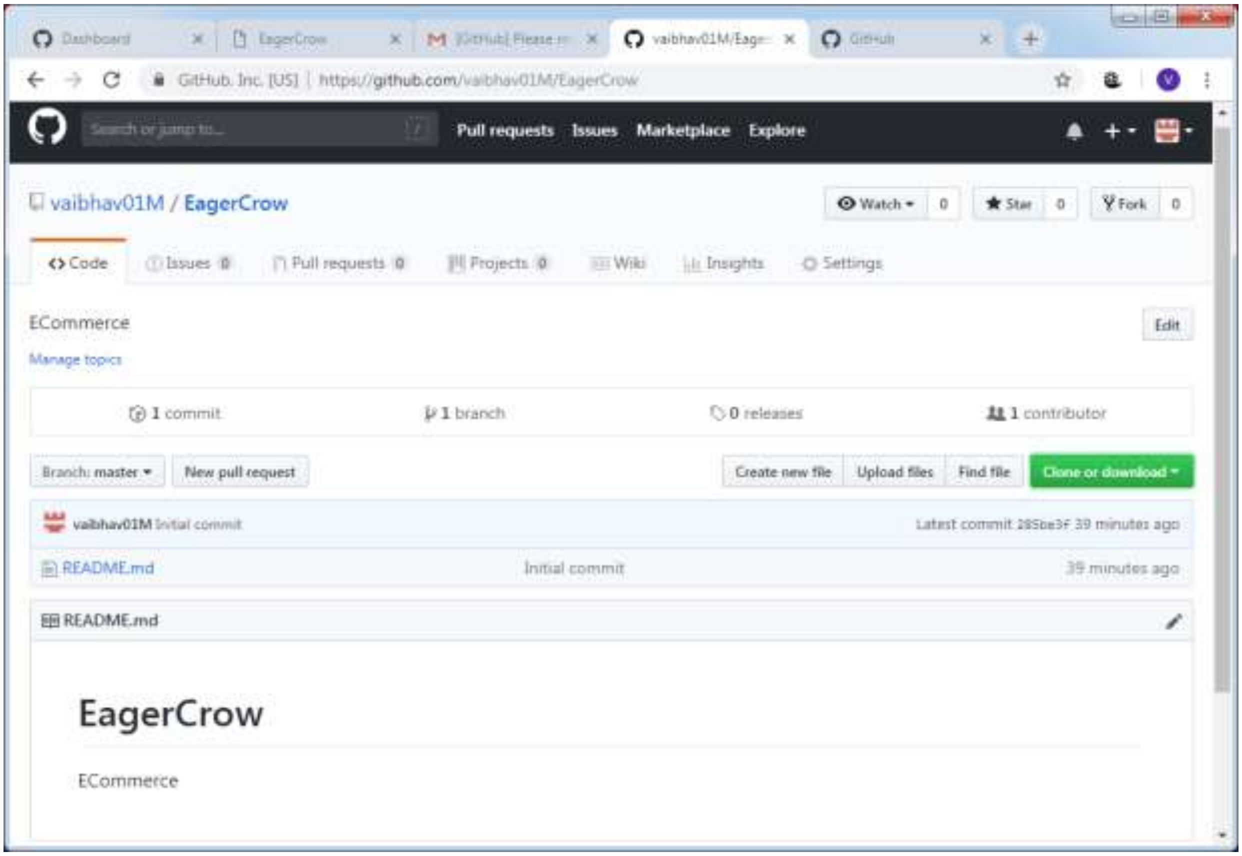


MINGW64:/f/Java Basic/GitEagerCrowLocalRepository

```
User@User-PC MINGW64 /f/Java Basic/GitEagerCrowLocalRepository
$ git init
Initialized empty Git repository in F:/Java Basic/GitEagerCrowLocalRepository/.g
it/

User@User-PC MINGW64 /f/Java Basic/GitEagerCrowLocalRepository (master)
$
```

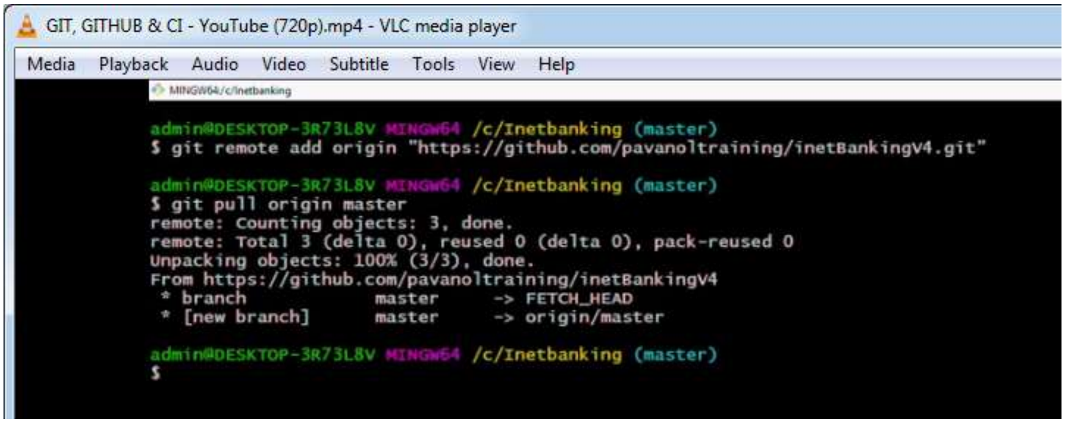
Actions -

1. Pulling Of file from **GitHub** (reomte) to **Git** (local)



Our GitHub remote repository having README file. Refer above picture. We have to **pull** that file to our local Repository Git.

Command : $ git pull origin master   and press Enter button


GIT, GITHUB & CI - YouTube (720p).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
MINGW64:/c/Inetbanking

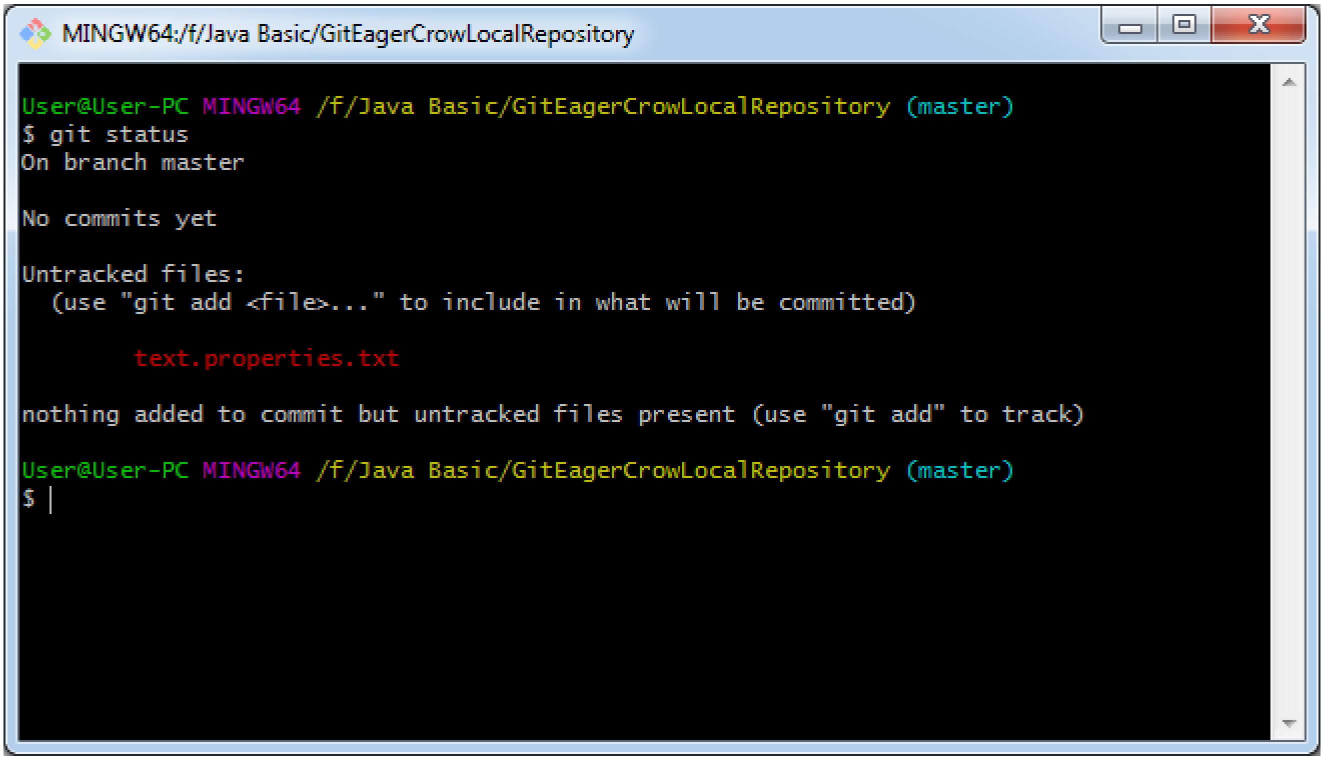
```
admin@DESKTOP-3R73L8V MINGW64 /c/Inetbanking (master)
$ git remote add origin "https://github.com/pavanoltraining/inetBankingV4.git"

admin@DESKTOP-3R73L8V MINGW64 /c/Inetbanking (master)
$ git pull origin master
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/pavanoltraining/inetBankingV4
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> origin/master

admin@DESKTOP-3R73L8V MINGW64 /c/Inetbanking (master)
$
```

This pic is of SDET videos pic not practice pic

After executing above command, we will see readme file will be imported to your Loacl Git repository.

2. If number of files are present in a folder It doesn’t mean that, that all files are part of Git repository (local repository). May be some files are part of Git and some files are not. Then how to check this status of local repository i.e. Git. Use following command ..

   $ git status   and press enter



```
User@User-PC MINGW64 /f/Java Basic/GitEagerCrowLocalRepository (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        text.properties.txt

nothing added to commit but untracked files present (use "git add" to track)

User@User-PC MINGW64 /f/Java Basic/GitEagerCrowLocalRepository (master)
$ |
```

Above pics shows, text.properties is file present in a folder but not part of Git

3. adding a file from **folder** to **Git** and to **GitHub**

$ git add (file name)

i.e. $ git add text.properties   press enter (to add specific file to index)

$ git add -a press enter (to add all unadded files to index)

Note : files which are added to index are ready to add in Git Repository (local)

4. adding file to Git repository

   $ git commit -m “---- some comment (msg)------” press Enter .

   (the file **text.properties** is get added to Git Repository we can check status)

5. how to add multiple file to GitHub

   After addition of files to repository folder and modification of already added file it is necessary to add that files to index first then it will be ready to add into **Git** local Repository by “ **$ git commit (file name)** ”.

   $ git add –A press Enter

   (above command will add all file to index)
   Then these all files are ready commit to Git repository by using commit command.

   $ gir commit –a –m “-----commment-------” press enter
   (above command will add files with comment)

6. How to see log of action on a Git Repository

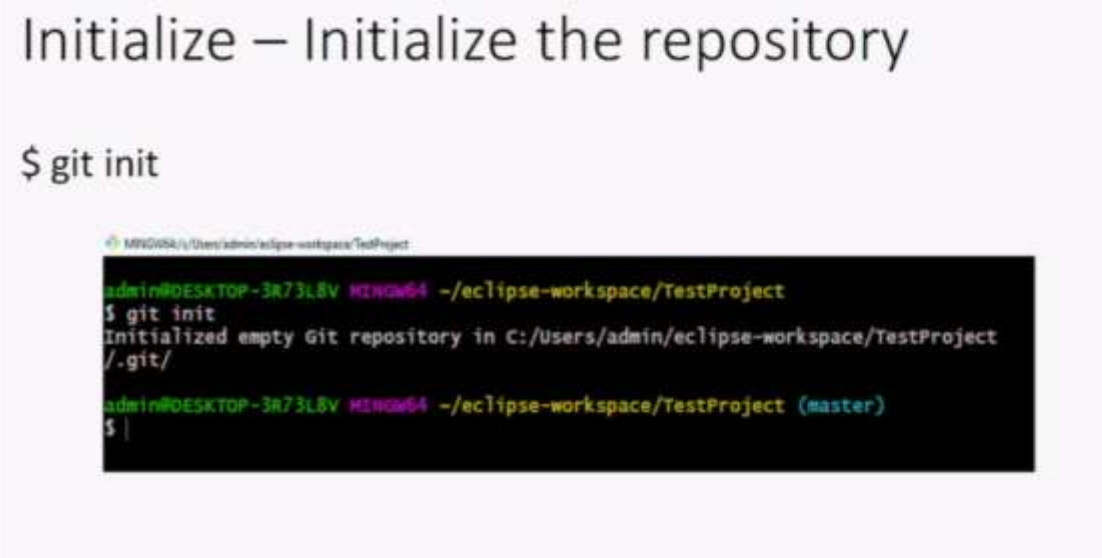
   $ git log presss enter (this will show you actions log)

7. File transfer from Local **Git** to Remote **Git Hub**

   $ git push origin master press enter

   (the files of local Git Repo will be added to remote GitHub termed that GitHub Repo files are unchanged , unmodified and same as it was means no new files are added to GitHub. If not then first we have to pull that file first then we can push tp Git Hub)

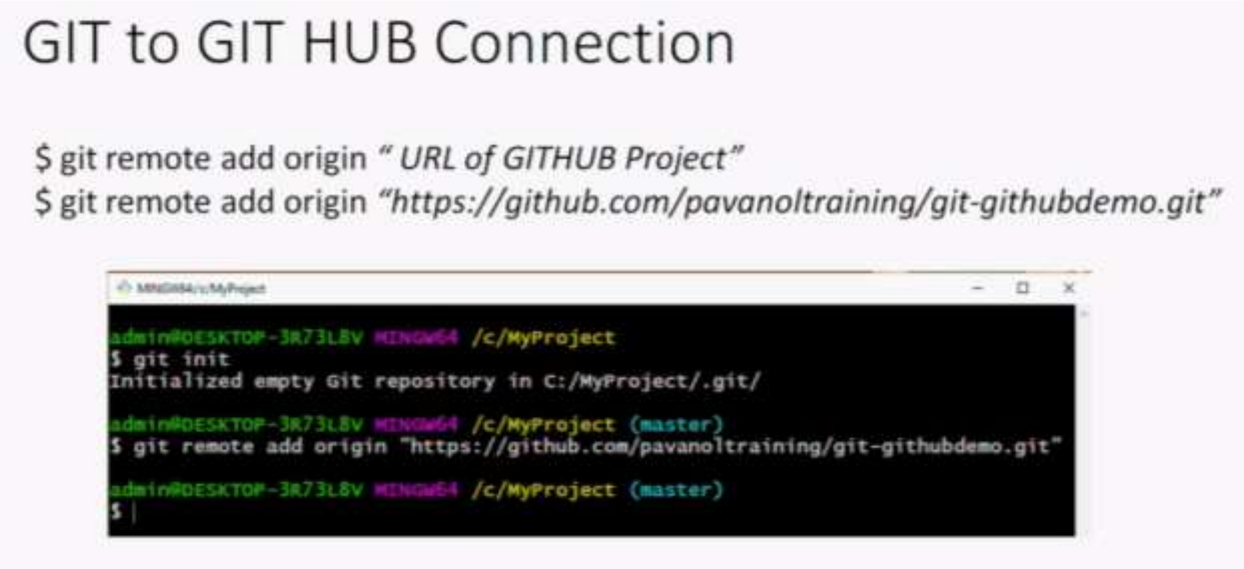
Over View of Commands



Initialize – Initialize the repository

$ git init

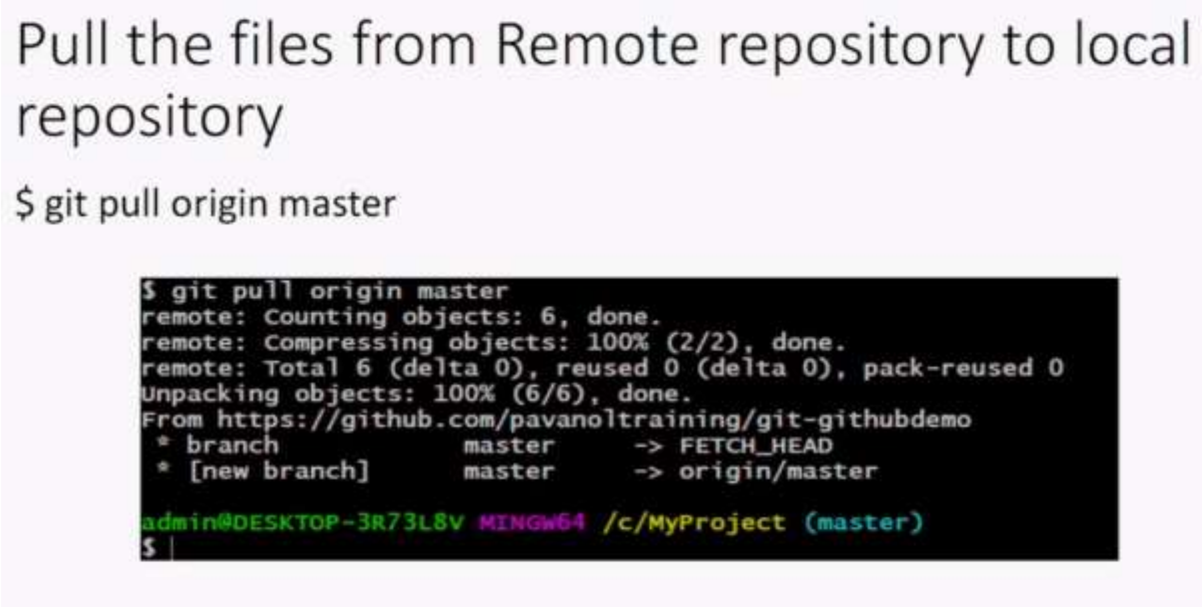
```
admin@DESKTOP-3R73L8V MINGW64 ~/eclipse-workspace/TestProject
$ git init
Initialized empty Git repository in C:/Users/admin/eclipse-workspace/TestProject
/.git/

admin@DESKTOP-3R73L8V MINGW64 ~/eclipse-workspace/TestProject (master)
$ |
```



GIT to GIT HUB Connection

$ git remote add origin *" URL of GITHUB Project"*
$ git remote add origin *"https://github.com/pavanoltraining/git-githubdemo.git"*

```
admin@DESKTOP-3R73L8V MINGW64 /c/MyProject
$ git init
Initialized empty Git repository in C:/MyProject/.git/

admin@DESKTOP-3R73L8V MINGW64 /c/MyProject (master)
$ git remote add origin "https://github.com/pavanoltraining/git-githubdemo.git"

admin@DESKTOP-3R73L8V MINGW64 /c/MyProject (master)
$ |
```



Pull the files from Remote repository to local repository

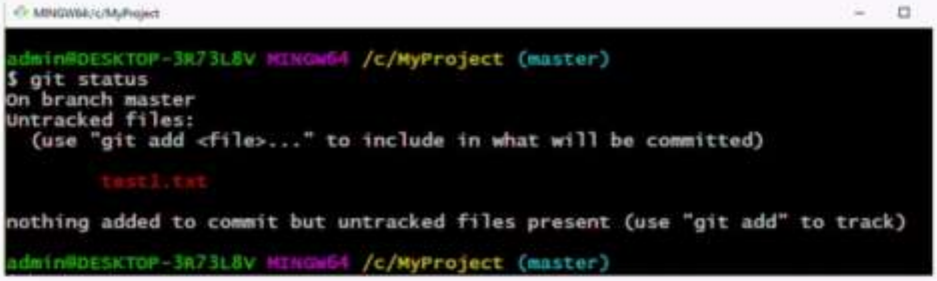
$ git pull origin master

```
$ git pull origin master
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
From https://github.com/pavanoltraining/git-githubdemo
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> origin/master

admin@DESKTOP-3R73L8V MINGW64 /c/MyProject (master)
$ |
```

Lists all the modified/created files which are ready to be added to the local repository.

$ git status

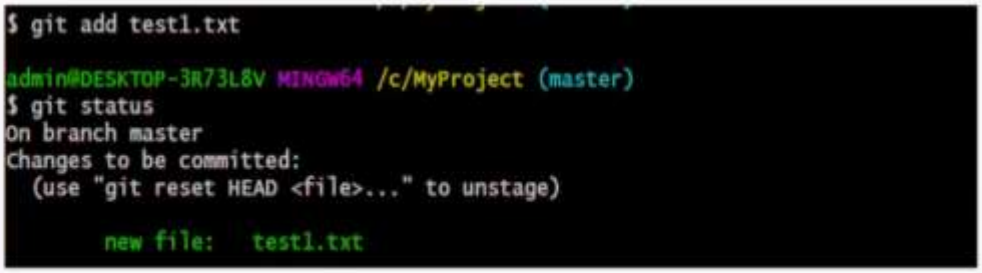
**create a file in local GIT repository folder (test1.txt). Then check status



```
admin@DESKTOP-3R73L8V MINGW64 /c/MyProject (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test1.txt

nothing added to commit but untracked files present (use "git add" to track)

admin@DESKTOP-3R73L8V MINGW64 /c/MyProject (master)
```

Add File to Index before commiting

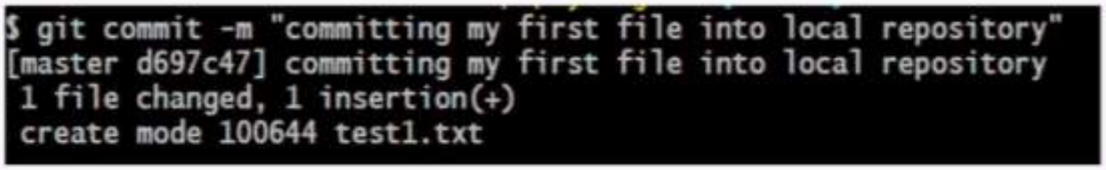
$ git add test1.txt



```
$ git add test1.txt

admin@DESKTOP-3R73L8V MINGW64 /c/MyProject (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   test1.txt
```

Committing single file into GIT repository.

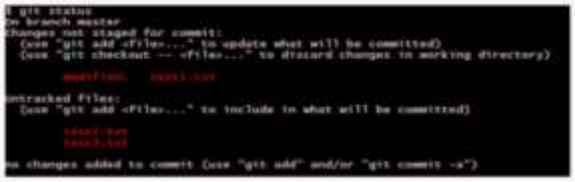
- Before committing , first time you need to add user.
- Below commands need to execute.
- git config --global user.name "pavan"
- git config --global user.email pavanoltraining@gmail.com



```
$ git commit -m "committing my first file into local repository"
[master d697c47] committing my first file into local repository
 1 file changed, 1 insertion(+)
 create mode 100644 test1.txt
```

Commiting multiple files

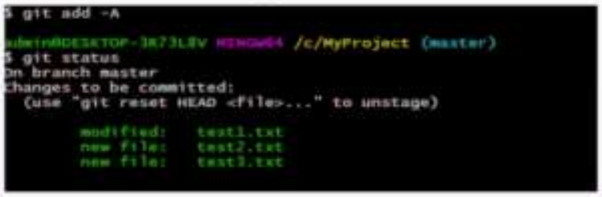
- Create few more files in local GIT repository folder (test2.txt,test3.txt) And also modify existing file test1.txt. Then check status.



```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   test1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test2.txt
        test3.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

- Add all files to index and check status again.

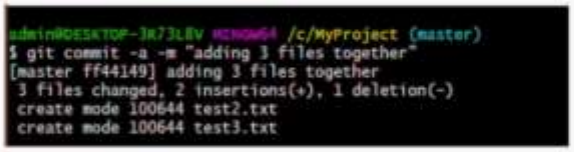
$ add -A



```
$ git add -A

admin@DESKTOP-3R73L8V MINGW64 /c/MyProject (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   test1.txt
        new file:   test2.txt
        new file:   test3.txt
```

- Commit all the files now.

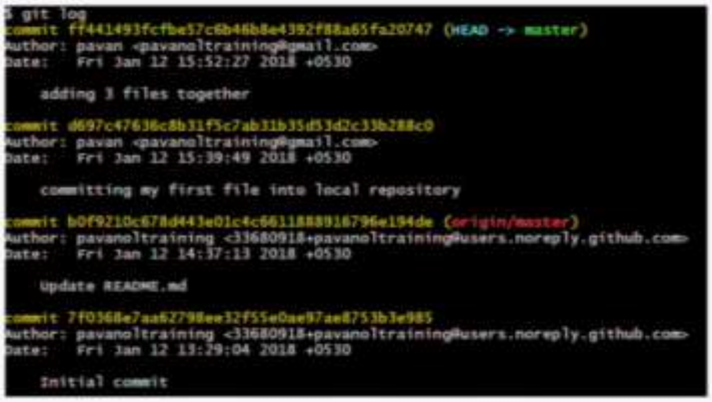
$ git commit -a -m "adding all the files"



```
admin@DESKTOP-3R73L8V MINGW64 /c/MyProject (master)
$ git commit -a -m "adding 3 files together"
[master ff44149] adding 3 files together
 3 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 test2.txt
 create mode 100644 test3.txt
```

See the all the commits in log

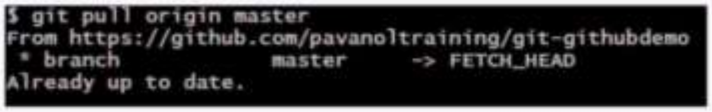
$ git log



```
$ git log
commit ff441493fcfbe57c6b46b8e4392f88a65fa20747 (HEAD -> master)
Author: pavan <pavanoltraining@gmail.com>
Date:   Fri Jan 12 15:52:27 2018 +0530

    adding 3 files together

commit d697c47636c8b31f5c7ab31b35d53d2c33b288c0
Author: pavan <pavanoltraining@gmail.com>
Date:   Fri Jan 12 15:39:49 2018 +0530

    committing my first file into local repository

commit b0f9210c678d443e01c4c6611888916796e194de (origin/master)
Author: pavanoltraining <33680918+pavanoltraining@users.noreply.github.com>
Date:   Fri Jan 12 14:37:13 2018 +0530

    Update README.md

commit 7f0368e7aa62798ee32f55e0ae97ae8753b3e985
Author: pavanoltraining <33680918+pavanoltraining@users.noreply.github.com>
Date:   Fri Jan 12 13:29:04 2018 +0530

    Initial commit
```

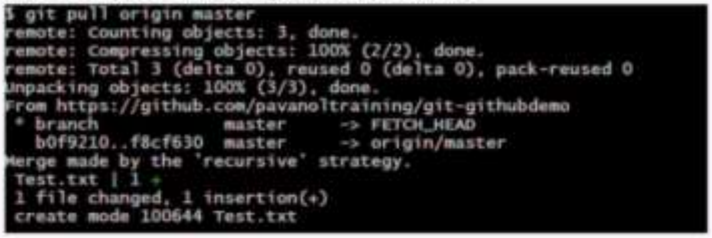
Pulling Files from Remote to Local

- The **git pull** command fetches changes from a remote repository to a local repository. It merges upstream changes in your local repository, which is a common task in GIT based collaborations.
- But first, you need to set your central repository as origin using the command:
- $ git pull origin master



```
$ git pull origin master
From https://github.com/pavanoltraining/git-githubdemo
 * branch            master     -> FETCH_HEAD
Already up to date.
```

- Try to upload one file in GITHUB directly for checking PULL operation.



```
$ git pull origin master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/pavanoltraining/git-githubdemo
 * branch            master     -> FETCH_HEAD
   b0f9210..f8cf630  master     -> origin/master
Merge made by the 'recursive' strategy.
 Test.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 Test.txt
```

Pushing Files from Remote to Local

$ git push origin master

Sr. No.	GIT Commands	Description
1	$ git ***init***	It initialize git local repository
2	$ git ***remote add origin*** “-url-”	It integrate git local & git remote Repository
3	$ git ***pull origin master***	It pulls file from remote local
4	$ git ***status***	It show the list of files which are part of git & which are not
5	$ git ***add***	It adds specific file to git
6	$ git ***add –a***	It adds all un added files
7	$ git ***commit –m “--msg--”***	It adds file to local repository with comment
8	$ git ***commit –a –m “—msg-- ”***	It adds all files to local repository with comment
9	$ git ***log***	It gives Action log
10	$ git ***push origin master***	It push code from local to remote repository
11	$ git ***config-global user.name***	It sets details of author like name & email id
12	$ git ***show[commit]***	It shows metadata & content changes of the specified commit
13	$ git ***diff***	It shows the file differences which are not yet staged.
14	$ git ***clone “—url—“***	It obtains repository from url
15	$ git ***tag***	It is used to give the tags to the specified commit
16	$ git ***merge***	It merges the specified branch History into current branch
17	$ git ***reset [file]***	It unstages the file, but it preserves the file contents
18	$ git ***branch***	It lists the all local branches of current repository
19	$ git ***checkout***	It is used to switch from one branch to another
20	$ git ***stash***	It temporary stores all modified tracked files

Manual Testing is a process of finding out the defects or bugs in a software program. In this method the tester plays an important role of end user and verifies that all the features of the application are working correctly. The tester **manually** executes **test** cases without using any automation tools.

The tester prepares a **test plan document** which describes the detailed and systematic approach to testing of software applications. Test cases are planned to cover almost 100% of the software application. As manual testing involves complete test cases it is a time consuming test.

The differences between actual and desired results are treated as **defects**. The defects are then fixed by the developer of software application. The tester retests the defects to ensure that defects are fixed. The goal of Manual testing is to ensure that application is defect & error free and is working fine to provide good quality work to customers.

Procedure of Manual Testing

1. Requirement Analysis
2. Test Plan Creation
3. Test case Creation
4. Test case Execution
5. Defect Logging
6. Defect Fix & Re-Verification
7.

Smoke Testing

Smoke Testing is a kind of Software Testing performed after software build to ascertain that the critical functionalities of the program is working fine. It is executed "before" any detailed functional or regression tests are executed on the software build. The purpose is to reject a badly broken application, so that the QA team does not waste time installing and testing the software application.

In Smoke Testing, the test cases chosen cover the most important functionality or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system is working fine. For Example a typical smoke test would be - Verify that the application launches successfully, Check that the GUI is responsive ... etc.

Sanity Testing

Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to find that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.

The objective is "not" to verify thoroughly the new functionality, but to determine that the developer has applied some rationality (sanity) while producing the software.

For instance, if your scientific calculator gives the result of 2 + 2 =5! Then, there is no point testing the advanced functionalities like sin 30 + cos 50.

	Smoke Test	Sanity Test
	Smoke Testing is performed to find that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality / bugs have been fixed
Objective	to verify the "stability" of the system in order to proceed with more rigorous testing	to verify the "rationality" of the system in order to proceed with more rigorous testing
performed by	by developers or testers	by testers
Documentation	usually documented or scripted	usually not documented and is unscripted
	Smoke testing is a subset of Acceptance testing	Sanity testing is a subset of Regression Testing
Test Area	Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
	Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up
Also known as	Tester acceptance testing.	Build verification test.

Regression Testing

Regression Testing is testing to confirm that a recent program or code change has not adversely affected existing features. Regression Testing is nothing but full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that old code still works.

Need of Regression Testing

Regression Testing is required when there is a

- Change in requirements and code is modified according to the requirement
- New feature is added to the software
- Defect fixing
- Performance issue fix

Functional Testing

1. Adhoc Testing
2. Exploratory Testing
3. Globalization Testing
4. Integration Testing

STLC (Software Testing Life cycle)

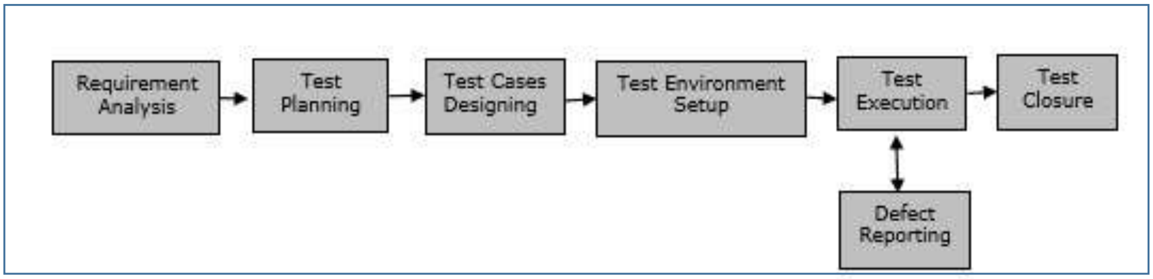
STLC stands for Software Testing Life Cycle. STLC is a sequence of different activities performed by the testing team to ensure the quality of the software or the product.

- STLC is an integral part of Software Development Life Cycle (SDLC). But, STLC deals only with the testing phases.
- STLC starts as soon as requirements are defined or SRD (Software Requirement Document) is shared by stakeholders.
- STLC provides a step-by-step process to ensure quality software.
- In the early stage of STLC, while the software or the product is developing, the tester can analyze and define the scope of testing, entry and exit criteria and also the Test Cases. It helps to reduce the test cycle time along with better quality.
- As soon as the development phase is over, the testers are ready with test cases and start with execution. This helps to find bugs in the initial phase.

STLC Phases

STLC has the following different phases but it is not mandatory to follow all phases. Phases are dependent on the nature of the software or the product, time and resources allocated for the testing and the model of SDLC that is to be followed.

STLC Diagram -

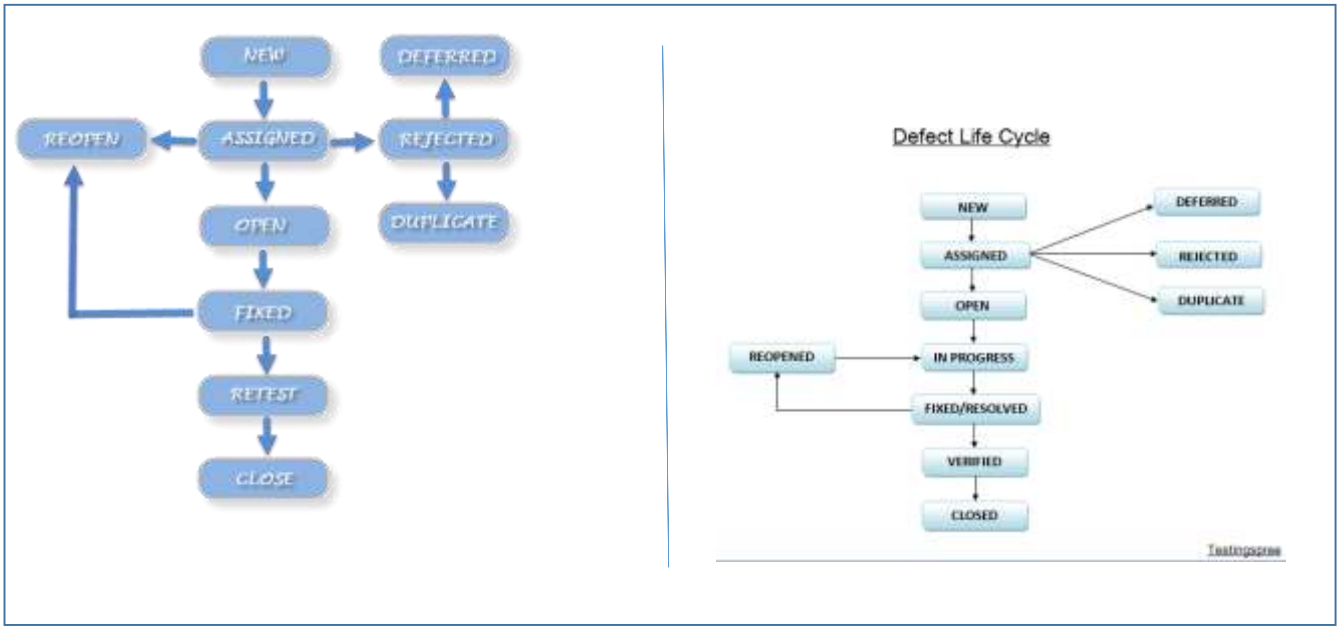


There are 6 major phases of STLC −

- **Requirement Analysis** − When the SRD is ready and shared with the stakeholders, the testing team starts high level analysis concerning the AUT (Application under Test).
- **Test Planning** − Test Team plans the strategy and approach.
- **Test Case Designing** − Develop the test cases based on scope and criteria's.

- **Test Environment Setup** − When integrated environment is ready to validate the product.
- **Test Execution** − Real-time validation of product and finding bugs.
- **Test Closure** − Once testing is completed, matrix, reports, results are documented.

DLC (Defect Life Cycle) -



Description -

New - If Bug found to a tester then it is assigned to a Developer team usually assignment task done by teamlead to Dev Team

1) **Assigned -** The assigned Bug is checked by Dev Team for its correctness (Wheather it is a Bug or not)

   - **Rejected –** If it is not a bug, it is rejected
   - **Deferred –** If it is a Bug, but not a Major bug or It may solved later on in a next sprint then it is deferred. Bug Status is deferred.
   - **Duplicated –** If the assigned bug is already logged by someone then on reassignment of that bug it is made as a Duplicated.

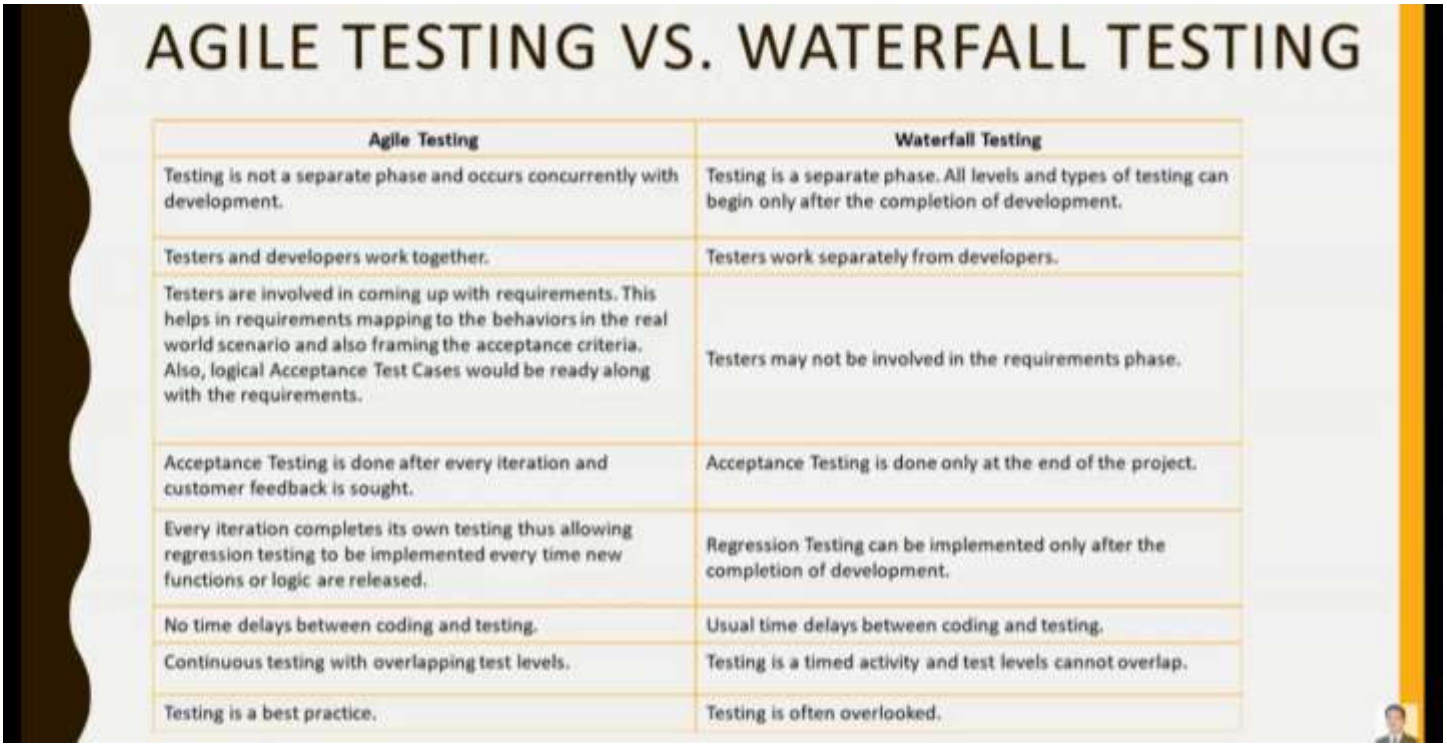
2) **Open –** Logged bug is opened by Dev Team

3) **Fixed -** After opening the bug is Fixed by Dev Team

   - **Reopen –** After Fixation of a bug it is reopened by Tester Team to check its proper Fixation

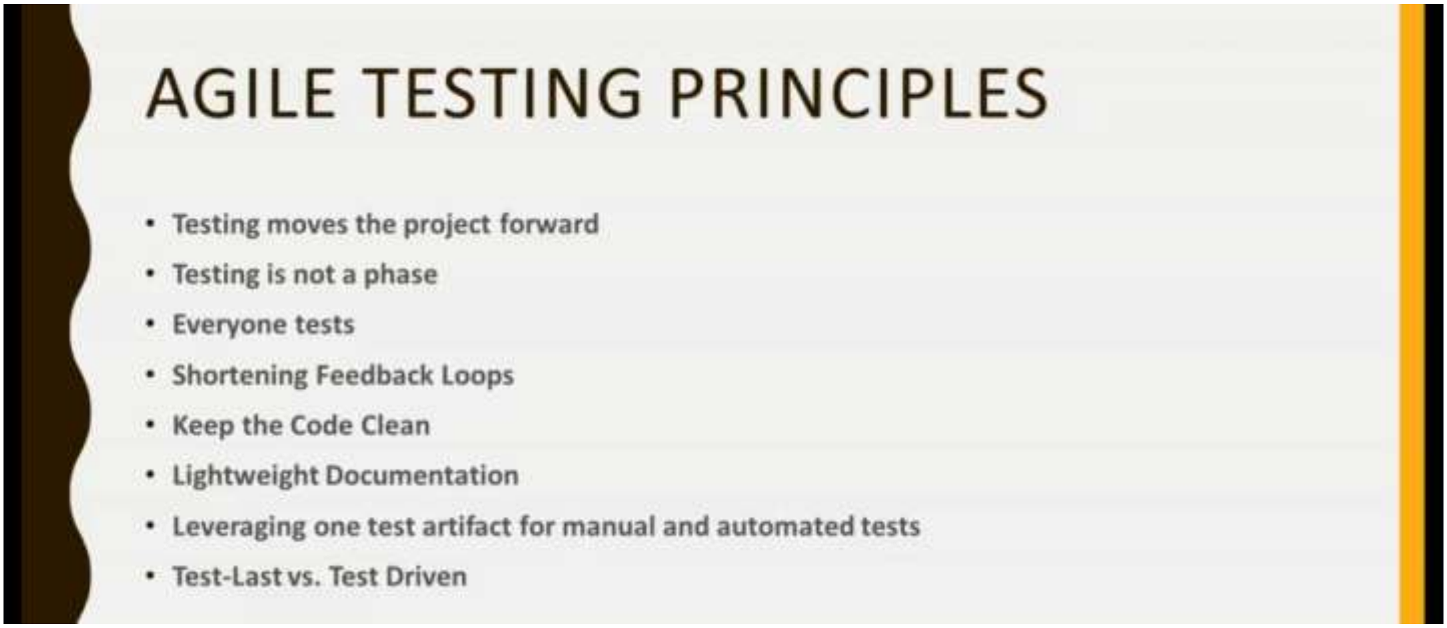
4) **Retest -** Fixed Bug is Retested to check wheather Bug is still exists or not. If the bug still exists then it is reassigned to Dev Team.

5) **Close –** After retesting the fixed bug not found again, means it is fixed properly then it is closed.



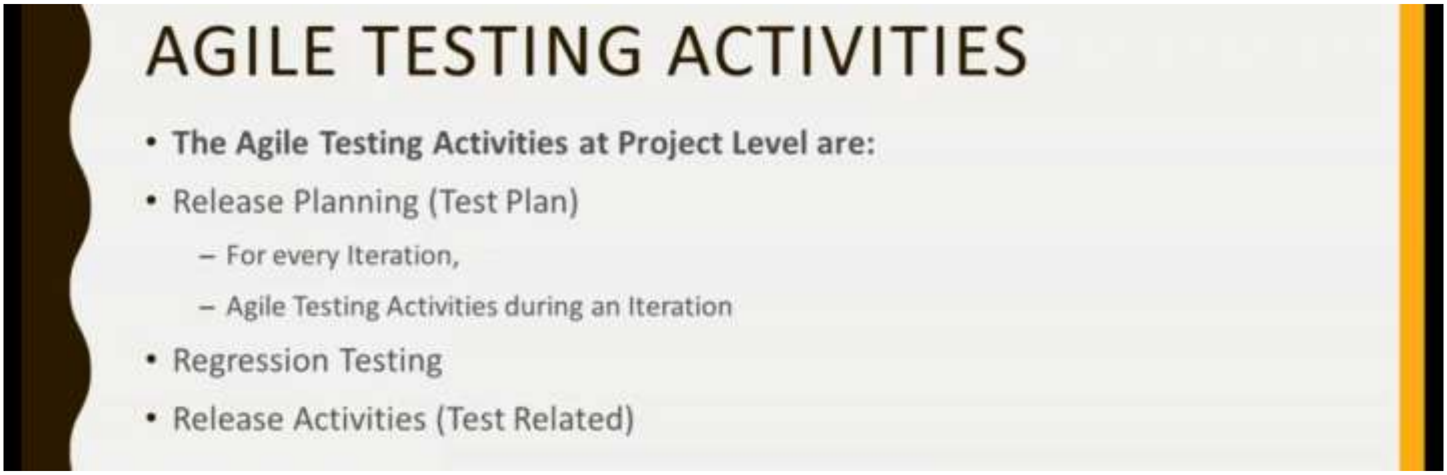
AGILE TESTING VS. WATERFALL TESTING

Agile Testing	Waterfall Testing
Testing is not a separate phase and occurs concurrently with development.	Testing is a separate phase. All levels and types of testing can begin only after the completion of development.
Testers and developers work together.	Testers work separately from developers.
Testers are involved in coming up with requirements. This helps in requirements mapping to the behaviors in the real world scenario and also framing the acceptance criteria. Also, logical Acceptance Test Cases would be ready along with the requirements.	Testers may not be involved in the requirements phase.
Acceptance Testing is done after every iteration and customer feedback is sought.	Acceptance Testing is done only at the end of the project.
Every iteration completes its own testing thus allowing regression testing to be implemented every time new functions or logic are released.	Regression Testing can be implemented only after the completion of development.
No time delays between coding and testing.	Usual time delays between coding and testing.
Continuous testing with overlapping test levels.	Testing is a timed activity and test levels cannot overlap.
Testing is a best practice.	Testing is often overlooked.



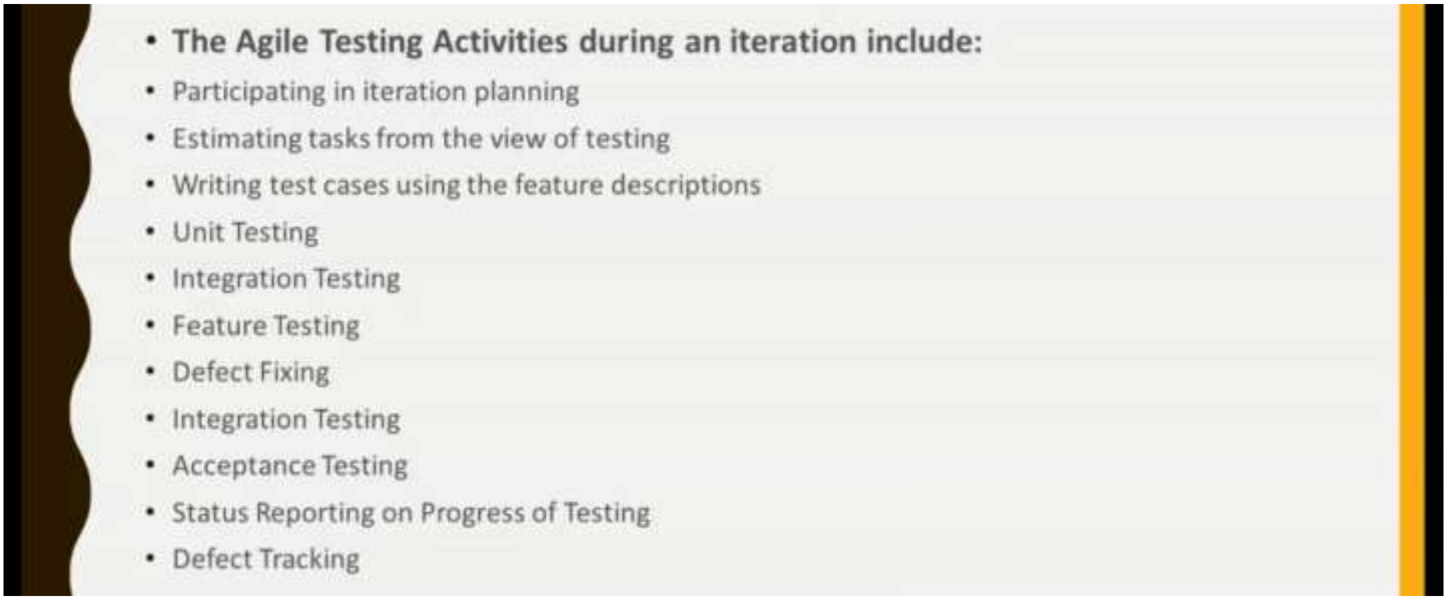
AGILE TESTING PRINCIPLES

- Testing moves the project forward
- Testing is not a phase
- Everyone tests
- Shortening Feedback Loops
- Keep the Code Clean
- Lightweight Documentation
- Leveraging one test artifact for manual and automated tests
- Test-Last vs. Test Driven



AGILE TESTING ACTIVITIES

- **The Agile Testing Activities at Project Level are:**
- Release Planning (Test Plan)
  - For every Iteration,
  - Agile Testing Activities during an Iteration
- Regression Testing
- Release Activities (Test Related)



- **The Agile Testing Activities during an iteration include:**
- Participating in iteration planning
- Estimating tasks from the view of testing
- Writing test cases using the feature descriptions
- Unit Testing
- Integration Testing
- Feature Testing
- Defect Fixing
- Integration Testing
- Acceptance Testing
- Status Reporting on Progress of Testing
- Defect Tracking

Every application in software industry maintains some log file. If you find any bug then we have to provide a steps, screenshots. Some times steps and screenshot are not enough to analyze the issue in that case Develpoment people will ask for log files.

Log4j Features

- It is thread-safe.
- It is optimized for speed.
- It is based on a named logger hierarchy.
- It supports multiple output appenders per logger.
- It supports internationalization.
- It is not restricted to a predefined set of facilities.
- Logging behavior can be set at runtime using a configuration file.
- It is designed to handle Java Exceptions from the start.
- It uses multiple levels, namely ***ALL, TRACE, DEBUG, INFO, WARN, ERROR*** and ***FATAL.***
- The format of the log output can be easily changed by extending the *Layout* class.
- The target of the log output as well as the writing strategy can be altered by implementations of the Appender interface.
- It is fail-stop. However, although it certainly strives to ensure delivery, ***log4j does not guarantee that each log statement will be delivered to its destination.***

Log4j has three main components:

- **loggers**: Responsible for capturing logging information.
- **appenders**: Responsible for publishing logging information to various preferred destinations. In simple words, it is used to write the logs in file. Following are few types of Appenders.
  1. ***Console Appender*** logs to standard output
  2. ***File appender*** prints logs to some file
  3. ***Rolling file appender*** to a file with maximum size
- **layouts**: Responsible for formatting logging information in different styles.

Log4j level: Primarily there are five kinds of log levels

1. All - This level of logging will log everything (it turns all the logs on)
2. DEBUG - print the debugging information and is helpful in development stage
3. INFO - print informational message that highlights the progress of the application
4. WARN - print information regarding faulty and unexpected system behavior.
5. ERROR - print error message that might allow system to continue
6. FATAL - print system critical information which are causing the application to crash
7. OFF - No logging

Configuring log4j involves assigning the Level, defining Appender, and specifying Layout objects in a configuration file. Add Log4j jar dependencies to your Maven project. These jar file are provided by Apache POI. We have to add either Log4j.xml file or Log4j.properties only to your project. We are adding Log4j.properties file to our demo project.

How log4j is configured?

To configure log4j we have to decide which appender to implement. Accordingly, parameters of appender will be set.

- We will use DEBUG level and ***RollingFileAppender***
- We will do two configurations or logs,
  - ***First***: root logger, that will write all system generated logs in file name i.e. ***Selenium.logs***
  - ***Second***: Will write the information generated by manual commands in code into the file name- ***Manual.logs***
- Layout will be ***PatternLayout***

#Root logger option

```
log4j.rootLogger=debug,file
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=F:\Java Basic\log4j\Logs\Selenium.logs
log4j.appender.file.maxfileSize=5000KB
log4j.appender.file.maxBackupIndex=1
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.conversionPattern=%d{ABSOLUTE} %5p %c<strong>{}</strong>:%L - %m%n
log4j.appender.file.Append=false
```

#Application Logs

```
log4j.logger.devpinoyLogger=DEBUG,dest1
log4j.appender.dest1=org.apache.log4j.RollingFileAppender
log4j.appender.dest1.maxfileSize=5000KB
log4j.appender.dest1.maxBackupIndex=3
log4j.appender.dest1.layout=org.apache.log4j.PatternLayout
log4j.appender.dest1conversionPattern=%d{dd/MM/yyyy HH:mm:ss} %c %m%n
log4j.appender.dest1.File=F:\Java Basic\log4j\Logs\Manual.logs
log4j.appender.dest1.Append=false
```

In the above Example, we have configured log4j to log in two different files named as Selenium.log and Manual.log.

- file and dest1 are the two identifiers.
- "File" is used to give file name in which logs will be saved
- "maxFileSize" is used to configure the maximum size of the log file. When file reaches this size, a new file will be created with the same name and the old file name will be add as an Index to it.
- "maxBackupIndex" is used to configure maximum number of files to be backup.
- "layout" is used to set the format of the log file.
- "Append" is used to set append function. If it is set to false, than every time a new file will be created rather than old file will be used for logging

Example Code –

```
import org.apache.log4j.Logger;
import org.apache.log4j.xml.DOMConfigurator;  // when using log4j.xml file
//  import org.apache.log4j.PropertyConfigurator; // when using log4j.properties file
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class Log4jDemo {
    public static void main(String[] args) {
      //  DOMConfigurator.configure("Log4j.xml");              // Configuring DOM Using  Log4j.xml file
      //  PropertyConfigurator.configure("Log4j.properties");  // When you are using property file
          System.setProperty("Webdriver.chrome.driver", "F:/chromedriver.exe");
          WebDriver driver = new ChromeDriver();
       Logger logger = Logger.getLogger("devpinoyLogger");    // Creating Logger Object
          driver.manage().window().maximize();
      logger.info("Browesr Launched");
          driver.get("https://www.facebook.com");
      logger.info("url entered");
          driver.findElement(By.xpath("//input[@type='email']")).sendKeys("abc@abc.com");
      logger.info("Email ID Entered");
          driver.findElement(By.id("pass")).sendKeys("xyz");
      logger.info("Password Entered");
          driver.findElement(By.id("u_0_2")).click();
      logger.info("Clicked on LogIN Button");
          }
}
```

Creating a logger object -

```
Logger logger = Logger.getLogger("class Name");
DOMConfigurator.configure("Log4j.xml"); // when you are using xml file
System.setProperty("webdriver.chrome.driver","---path--- ");
```

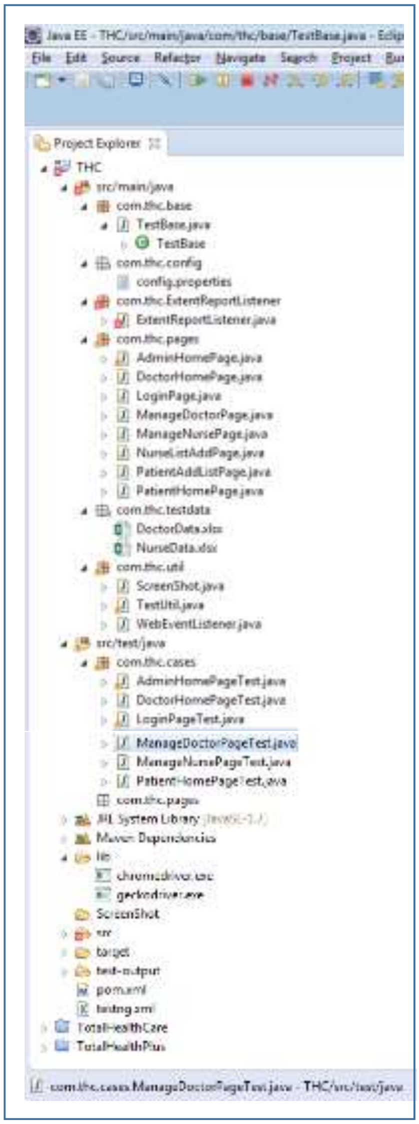
```
Logger logger = Logger.getLogger("class Name");
PropertyConfigurator.configure("Log4j.properties"); // when you are using properties file
System.setProperty("webdriver.chrome.driver","---path--- ");
```

How To Explain Selenium Framework (Notes By – Vaibhav Mhaskar)

Project Name **: Total Health Plus**

Domain **: Health care**

Framework **: Hybrid= Data Driven + TestNg + Maven + log4j**


Java EE - THC/src/main/java/com/thc/base/TestBase.java - Eclip
File Edit Source Refactor Navigate Search Project Run
Project Explorer

```
THC
  src/main/java
    com.thc.base
      TestBase.java
        TestBase
    com.thc.config
      config.properties
    com.thc.ExtentReportListener
      ExtentReportListener.java
    com.thc.pages
      AdminHomePage.java
      DoctorHomePage.java
      LoginPage.java
      ManageDoctorPage.java
      ManageNursePage.java
      NurseListAddPage.java
      PatientAddListPage.java
      PatientHomePage.java
    com.thc.testdata
      DoctorData.xlsx
      NurseData.xlsx
    com.thc.util
      ScreenShot.java
      TestUtil.java
      WebEventListener.java
  src/test/java
    com.thc.cases
      AdminHomePageTest.java
      DoctorHomePageTest.java
      LoginPageTest.java
      ManageDoctorPageTest.java
      ManageNursePageTest.java
      PatientHomePageTest.java
    com.thc.pages
  JRE System Library [JavaSE-1.7]
  Maven Dependencies
  lib
    chromedriver.exe
    geckodriver.exe
  ScreenShot
  src
  target
  test-output
  pom.xml
  testng.xml
TotalHealthCare
TotalHealthPlus
com.thc.cases.ManageDoctorPageTest.java - THC/src/test/java
```

Packages –

Com.thc.base → Env property file reading, Browser Launch

Com.thc.config → Env Prop file includes browser name, userid, pasword

Com.thc.reportListener → contains class which extends IReporter Interface

Com.thc.pages→ web Pages with page Elements initialized in Pagefactory

Com.thc.testdata→ excel file which includes different types of test data

Com.thc.testCases → test cases

Com.thc.util→ common utilitis ie. Excel file reading, ScreenShot code, Event Listener code

Maven Dependencies→ jars dependencies

Lib Folder → Browser Drivers

ScreenShot Folder → to store screenshots

test-output → testng.xml

pom.xml → maven dependencies

Description

1	**Coding Language**	Java
3	**Test Base Class**	common functions used by all classes. This class is responsible for loading the files Initializing webdriver, impicite waits, extent report, it has fis object to indicate the file from which data to be read.
4	**Utility Class**	repetitive code taking screen shots, excel sheet reading this class extends the TestBase class and inherites the properties of TestBase class to Utils
5	**Test Data**	include excel file of Testdata, we have used Apache POI to handle Excel sheet
6	**Page Object Model**	Framework designing approach
7	**Functions**	
8	**Property Files**	this file include browser name, Application URL, path to store screen shots, login credential UserID, password etc
9	**TestNG**	which kind of annotations are used ? we have used TestNG for assertion, grouping and parallel execution
10	**Parameterization**	Excel File
11	**Error Screenshots**	how u have taken screen shots using Ashot
12	**Sending emails**	
13	**Reports**	***extent*** reports, mainting logs, shows screenshots of failed test cases
14	**Version Conrtol used**	***Git*** and ***Git Hub*** to store test scripts
15	**Continuous Integration tool**	***Jenkins*** - integrating testNg dependency in pom.xml file and running this pom.xml file using Jenkins and also for nightly execution based on schedule.
16	**Project building tool**	we are using ***Maven*** for build execution and dependencies purpose
17	**Log**	***log4j*** used for log report

Some question asked in

1) Where you have used Inheritance in your framework?
2) Where you have used Data Hiding in your framework?
3) Where you have used Method Overloading in your framework?
4) Where you have used Method OverRiding in your framework?
5) Where you have used Method Polymorphism in your framework?
6) Where you have used Method abstraction in your framework?
7) Where you have used Method Encapsulation in your framework?

Dynamic WebTable Handling (Notes By – Vaibhav Mhaskar)

There are two types of Web Table- Static web Table and Dynamic WebTable. The Table whose content are not changing is a ***Static WebTables*** and the table whose contents are changing frequently after some time duration is the ***Dynamic WebTable***.

XML code of WebTable –

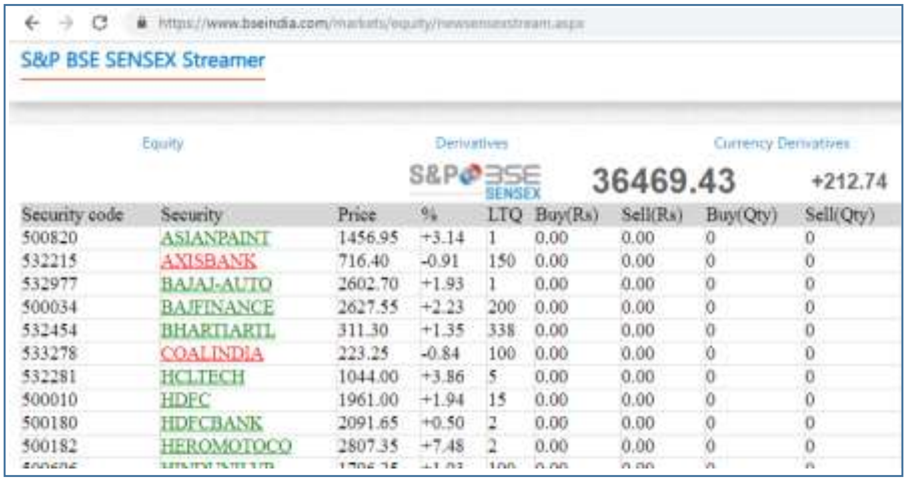
```
<tabel>
   <tbody>
          <tr>
               <td> Sr.No </td>
               <td> Student Name </td>
               <td> Roll Number </td>
         </tr>
         <tr>
               <td id="firstStudent">001</td >
               <td>Avinash</td >
               <td>A100</td >
        </tr>
    </tbody>
</table>
```

Static WebTable Handling –

```
public class WebTableHandling {
 public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver","\\lib\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.w3schools.com/html/html_tables.asp");  //Check for Example on w3school
        driver.manage().window().maximize();
        List allRows = driver.findElements(By.xpath("//table[@id='customers']/tbody/tr"));
        System.out.println("Total Number of rows are : " + allRows.size());
                 System.out.println("------------------------ Starting Table ---------------------");
                for (int row = 0; row < allRows.size(); row++) {
                  System.out.println(((WebElement) allRows.get(row)).getText());
                }
                 System.out.println("------------------------ Ending Table ----------------------");
        List  allCols = driver.findElements(By.xpath("//table[@id='customers']/tbody/tr[2]/td"));
        System.out.println("Total Number of Coloumns are : " + allCols.size());
                for (int i = 0; i < allCols.size(); i++) {
                  System.out.println(((WebElement) allCols.get(i)).getText());
                }
        WebElement row = driver.findElement(By.xpath("//table[@id='customers']/tbody/tr[5]"));
        System.out.println(row.getText());
        }}
```

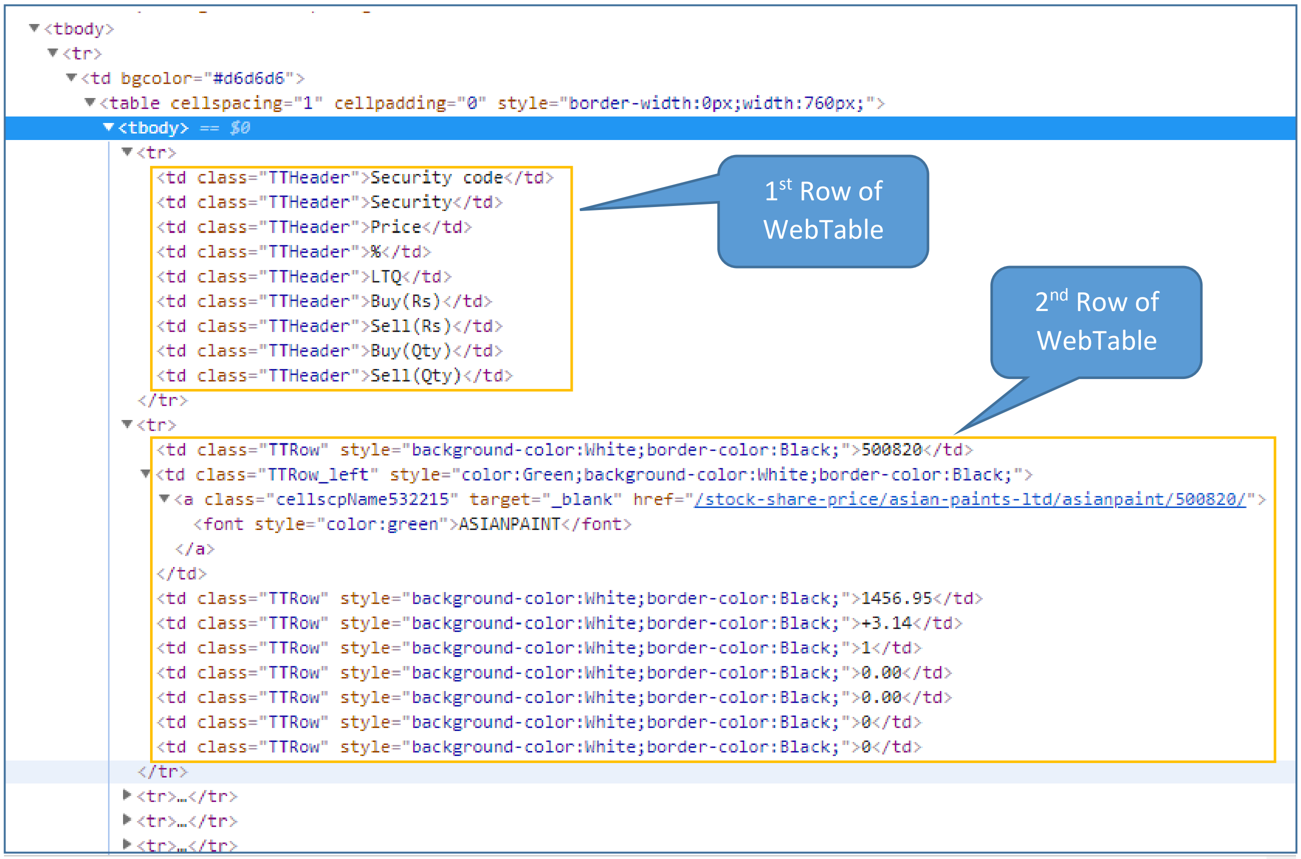
Dynamic WebTable Handling –

Load the following url – *“www.bseindia.com/markets/equity/newsensexstream.aspx”* and inspect the table.

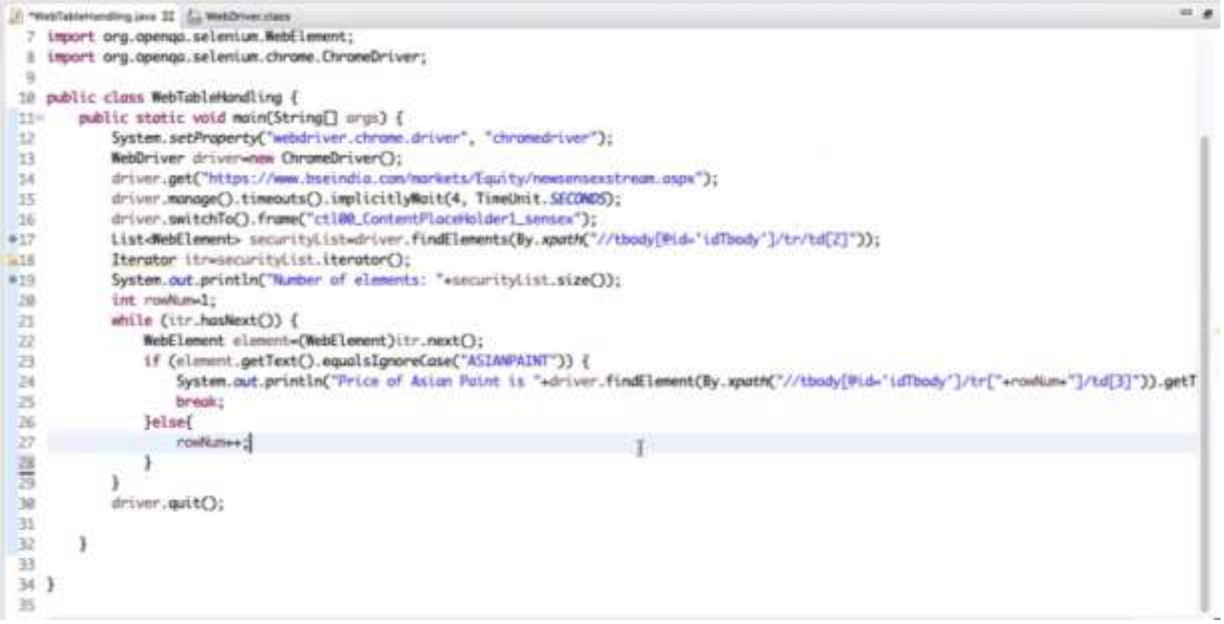


36469.43 +212.74

Security code	Security	Price	%	LTQ	Buy(Rs)	Sell(Rs)	Buy(Qty)	Sell(Qty)
500820	ASIANPAINT	1456.95	+3.14	1	0.00	0.00	0	0
532215	AXISBANK	716.40	-0.91	150	0.00	0.00	0	0
532977	BAJAJ-AUTO	2602.70	+1.93	1	0.00	0.00	0	0
500034	BAJFINANCE	2627.55	+2.23	200	0.00	0.00	0	0
532454	BHARTIARTL	311.30	+1.35	338	0.00	0.00	0	0
533278	COALINDIA	223.25	-0.84	100	0.00	0.00	0	0
532281	HCLTECH	1044.00	+3.86	5	0.00	0.00	0	0
500010	HDFC	1961.00	+1.94	15	0.00	0.00	0	0
500180	HDFCBANK	2091.65	+0.50	2	0.00	0.00	0	0
500182	HEROMOTOCO	2807.35	+7.48	2	0.00	0.00	0	0
500696	HINDUNILVR	1796.25	+1.03	100	0.00	0.00	0	0



```
▼<tbody>
  ▼<tr>
    ▼<td bgcolor="#d6d6d6">
      ▼<table cellspacing="1" cellpadding="0" style="border-width:0px;width:760px;">
        ▼<tbody> == $0
          ▼<tr>
              <td class="TTHeader">Security code</td>
              <td class="TTHeader">Security</td>
              <td class="TTHeader">Price</td>
              <td class="TTHeader">%</td>
              <td class="TTHeader">LTQ</td>
              <td class="TTHeader">Buy(Rs)</td>
              <td class="TTHeader">Sell(Rs)</td>
              <td class="TTHeader">Buy(Qty)</td>
              <td class="TTHeader">Sell(Qty)</td>
            </tr>
          ▼<tr>
              <td class="TTRow" style="background-color:White;border-color:Black;">500820</td>
            ▼<td class="TTRow_left" style="color:Green;background-color:White;border-color:Black;">
              ▼<a class="cellscpName532215" target="_blank" href="/stock-share-price/asian-paints-ltd/asianpaint/500820/">
                  <font style="color:green">ASIANPAINT</font>
                </a>
              </td>
              <td class="TTRow" style="background-color:White;border-color:Black;">1456.95</td>
              <td class="TTRow" style="background-color:White;border-color:Black;">+3.14</td>
              <td class="TTRow" style="background-color:White;border-color:Black;">1</td>
              <td class="TTRow" style="background-color:White;border-color:Black;">0.00</td>
              <td class="TTRow" style="background-color:White;border-color:Black;">0.00</td>
              <td class="TTRow" style="background-color:White;border-color:Black;">0</td>
              <td class="TTRow" style="background-color:White;border-color:Black;">0</td>
            </tr>
          ►<tr>…</tr>
          ►<tr>…</tr>
          ►<tr>…</tr>
```



```
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class WebTableHandling {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "chromedriver");
        WebDriver driver=new ChromeDriver();
        driver.get("https://www.bseindia.com/markets/Equity/newsensexstream.aspx");
        driver.manage().timeouts().implicitlyWait(4, TimeUnit.SECONDS);
        driver.switchTo().frame("ctl00_ContentPlaceHolder1_sensex");
        List<WebElement> securityList=driver.findElements(By.xpath("//tbody[@id='idTbody']/tr/td[2]"));
        Iterator itr=securityList.iterator();
        System.out.println("Number of elements: "+securityList.size());
        int rowNum=1;
        while (itr.hasNext()) {
            WebElement element=(WebElement)itr.next();
            if (element.getText().equalsIgnoreCase("ASIANPAINT")) {
                System.out.println("Price of Asian Paint is "+driver.findElement(By.xpath("//tbody[@id='idTbody']/tr["+rowNum+"]/td[3]")).getT
                break;
            }else{
                rowNum++;
            }
        }
        driver.quit();

    }

}
```

(Above code did by Avi sir… search on youtube)

Software/ plugin / jars Versions

Software / Plugin / Jar	**Version Used**	**Available Latest Version**
Java	7	8
Eclipse	Mars.2, 4.5.2	
TestNG	6.14.3	7.0.0 beta3
JRE SE 1.7	SE1.7	
Selenium – java	**3.6.0**	3.141.59
Apache POI	**3.17**	4.0.1
Log4j	1.2.17	2.11.1
Ashot	1.5.4	1.5.4
Junit	3.8.1 / 4.12	5.4.0
extentreports	2.41.2	2.41.2
Git Git Hub	5.2.0	5.2.0
Mantiss (Bug Tracking Tool)		
QC (Bug Track & Test Case management Tool)		

1) What is Maven?

Maven is a project management tool. It is based on POM (Project Object Model).

2) What aspects are managed by Maven?

- Builds
- Documentation
- Reporting
- SCMs
- Releases
- Distribution

3) What are the advantages of Maven?

- No need to add jar file in each project
- Creates right directory structure
- Builds and deploys the project

4) What is the command to check the maven version?

Type the following command on console to know the maven version. **- mvn -version**

5) What does the build tool?

- Generates source code (if the auto-generated code is used)
- Generates documentation from source code
- Compiles source code
- Packages compiled code into a JAR or ZIP file
- Installs the packaged code in the local repository, server repository, or central repository

6) What is the difference between Ant and Maven?

Ant	Maven
It is a tool box	It is Framework
It is mainly a build tool	It is project Management toll
There is no **life cycle**.	There is a **life cycle**.
Ant **doesn't have formal conventions**.	Maven **has a convention** to place source code, compiled code etc.
Ant is **procedural**.	Maven is **declarative**.
The ant scripts are **not reusable**.	The Maven plugins are **reusable**.

7) Why is the use of the profile required in Maven?

For providing probability to projects, we use profiles.

8) What is the syntax for offline project creation?

The syntax for project creation is: mvn o packg.

9) How is the propagation of plugins to child POMs stopped?

It can be done using the following syntax: set**<inherited>** to false.

10) What is the use of the exclusion element?

The element is used to exclude dependencies.

11) Define SNAPSHOT in terms of maven.

The snapshot indicates the current development copy.

12) Define Archetype.

It is a Maven plugin which is designed for the creation of project structure.

13) Give the command for installation of the JAR file in a local repository.

mvn install

14) Mention the phases of cleaning lifecycle.

The lifecycle of cleaning consist of:

- pre-clean
- clean
- post-clean

15) What is the purpose of mvn clean command?

The command removes the target directory before the starting of a build process.

16) What is a MOJO?

A MOJO stands for Maven plain Old Java Object. Each MOJO is an executable goal in Maven, and a plugin is a distribution of one or more related MOJOs.

17) What is a repository?

A repository is a directory or place where all the jars and pom.xml file are stored. There are 3 types of a repository in Maven:

1. Local Repository
2. Central Repository
3. Remote Repository

18) What is a local repository?

Maven local repository is created by maven in your local system when you run any maven command.

19) What is a central repository?

Maven community creates maven central repository on the web. More details.

20) What is a remote repository?

Maven remote repository is located on the web by different vendors. So you need to define the dependency in pom.xml file manually. It is important because most of the libraries are missing from the central repository.

21) What is POM?

POM stands for Project Object Model. The pom.xml file contains information of project and project configuration.

22) What are the build phases in Maven?

1. validate
2. compile
3. test
4. package
5. integration-test
6. verify
7. install
8. deploy

23) What is the command to package maven project?

mvn -package

24) What is the fully qualified artifact name of maven project?

<groupId>:**<artifactId>**:**<version>**

26) What are the phases of Maven?

Prepare Resource	Resource copying can be customized in this phase
Validate Phases	Validates everything is in order, configuration is running properly, the code is placed in a proper way etc.
Compiler phases	Source code is compiled in this phase
Test phases	Runs the test Cases which are specified for the code
Package phases	This package creates JAR / WAR packages as mentioned in the packaging in POM.xml file
Install phase	This phase installs the package in local / Remote Maven Repository.
Deploy phase	This phase copies the final package to remote repository

27) What are POM Files in Maven?

All your code and resources are placed in the src directory.

>> The main/Java directory holds your project code.

>> Compiled code is placed in the largest directory.

>> The test/Java directory holds your JUnit test code.

28) What is Maven Archetypes? And what are the Project Types?

Archetypes are templates to create a variety of Java project structures, including web applications specific to a container such as Wildfly. In other words, it is a tool that creates the stuff you build the project on top of.

Project Types:

1. War

2. Jar

3. Ear

4. Wildfly

5. JMS

6. Android

28) What is Maven Repository?

In Maven a repository is used as a storage folder or a directory to store your projects, your files such as Jar, War or Ear files that can be later used by the Maven application or tool. It works as a whole library of the files that is easily accessible and can be easily located in your system without any trouble and then can be used by Maven.

29) What are the archetype goals?

Four goals associated with archetype plugin:
Create - creates using a quick-start template.
Generate – provide a menu of templates.
Create-from-project – creates an archetype from an existing project.
Crawl – searches the repository for archetype and updates catalog.

30) What is Maven Artifact?

Maven Artifact consists of files like Jar file or War file that result in the expansion of the specific file in the Maven repository. The Jar file can use as an artifact in Maven. The Maven Artifact determined by a group ID name to run the Maven Artifact in the Maven. They can contain files like Ear, Jar, and War or Zip file as well. Maven Artifact is used for specifying applications to locate a name or package. It is usually stored in your system repository.

31) What is the sequence in which Maven searches for dependency libraries?

You can locate dependency in the local repository system of your software. Sometimes, it is difficult to locate or identify in the local repository. So, I can find or look in the central repository system and if it shows the dependency missing then one can look in remote repository to find the dependency. If it still shows the same thing repeatedly then the system will show error in finding the dependencies. And if the dependencies are found in the local repository then it will be automatically downloaded in the central repository for future use.

32) What are the things you need to define for each external dependency?

External Dependency plays an important part in the Maven software. It is an internal part of the system without which dependency cannot be located in a system. To specify the external dependency we need:

1. It requires a group ID duplicate to the library name.

2. It requires an artifact ID duplicate to the library name.

3. Mentioning of dependency scope in the system

4. Have to mention the system route corresponding to the project position.

33) What are the steps involved in project deployment?

There are several steps to follow while implying project deployment in Maven. These steps include:

1. Go through all the projects and analyze the code that is working in the background in progress in the source code repository and identifies it.

2. In order to get the project development, one need to download the whole source code from the Social Venture Network.
3. Construct or develop the application in the system
4. It needs to be saving as a War or Jar file system.
5. Get the specified file from the location path and move that specific file to create a site.
6. The application that is created in the system needs to be updated with the latest version with date and version number.

34) What are the aspects Maven manages?

1. Documentation
2. SCMs
3. Distribution
4. Builds
5. Reporting
6. Releases
7. Mailing list
8. Dependencies

35) What is a goal in Maven terminology?

The goals mentioned here in the Maven suggests the managing and building process requires creating a project. There is no limitation to follow the goals in Maven; it can build as many phases as it wants with zero boundations. You directly achieve your goal without any kind of outside intervention.

36) What kind of information does POM.xml file contains ?

POM contains the some of the following configuration information −

- project dependencies
- plugins
- goals
- build profiles
- project version
- developers
- mailing list

37) What is Maven Build Lifecycle?

A Build Lifecycle is a well defined sequence of phases which define the order in which the goals are to be executed. Here phase represents a stage in life cycle.

38) Name the 3 build lifecycle of Maven.

The three build lifecycles are −

- **clean:**cleans up artifacts created by prior builds.
- **default (or build):**This is used to build the application.
- **site:** generates site documentation for the project.

39) What is the command to quickly build your Maven site?

Type the command: mvn site

40) What would the command *mvn clean* do ?

This command removes the target directory with all the build data before starting the build process.

42) What is SNAPSHOT in Maven?

SNAPSHOT is a special version that indicates a current development copy. Unlike regular versions, Maven checks for a new SNAPSHOT version in a remote repository for every build.

43) How Maven handles and determines what version of dependency will be used when multiple version of an artifact are encountered?

Maven determines what version of a dependency is to be used when multiple versions of an artifact are encountered. If two dependency versions are at the same depth in the dependency tree, the first declared dependency will be used. This is called dependency mediation.

44) What is a Mojo?

A mojo is a Maven plain Old Java Object. Each mojo is an executable goal in Maven, and a plugin is a distribution of one or more related mojos.

45) What is a project's fully qualified artifact name?

<groupId>:<artifactId>:<version>

46) What is the command to create a new project based on an archtype?

Type the following command − mvn archetype : generate

Tell me About Your Self

Hi xxxxxx,

I am **XXXXXXXX** , having **N.N** years of experience as **a Q. A. Automation Tester**. I am working with XXXX from Jan/Feb/March 20XX.

I am experienced in (Domain Name)- ***Health Care*** and ***E-Commerce*** domain. I have worked on ***Selenium WebDriver*** in ***Eclipse IDE, Hybrid Driven Framework, TestNG***

My responsibilities are –

1. writing a Automation Testing Scripts using ***Java*** and ***selenium***
2. maintaining Automation framework which is ***Hybrid*** using ***POM***
3. Maintaing logs using ***Log4j***
4. ***Defect logging***
5. Generating customized report using ***Extent Report***
6. Understanding the BRS (Business Requirment Specification)
7. Attending Daily Scrum Meeting
8. Documentation
9. Involved in maintenance of Automation Framework
10. Developing scripts to read / write data from database

I am Engineering Graduated /Post Graduated in XXXX stream and currently living in Pune/XXXXXX

I am enthusiastic and dedicated to my work and looking for wonderful career in Automation Testing

(For this type of question you are supposed to explain your roles and responsibilities, duties, Key skills)