

# 內容

Part 1: Environment Setup .....	3
1. Java and Maven Setup.....	3
2. Eclipse Setting.....	3
Create a New Project.....	3
Maven Surefire Plugin .....	8
Part 2: Create Test Plan.....	12
Test Plan1-Create Open Web Browser.....	12
Test Plan2- Run Open Web Browser .....	18
Understand Locator.....	20
Test Plan3 Create login user and password page.....	21
Test Plan4 Incorrect username test .....	27
Test Plan5 Incorrect password test .....	29
Part 3: Creating Test Suite-Login Account.....	31
SetUP TestSuite-TestNG .....	31
Test Suite1 Run Login Test .....	34
Test Suite1-Run Login Success and Incorrect Account.....	34
TestingNG-Annotations:@Test-Priority .....	36
Login Incorrect account and password test .....	36
TestingNG-Annotations:@Test-Enable .....	38
Login Incorrect username Test.....	38
TestingNG-Annotations:@Test Groups.....	39
Login Incorrect username Test.....	39
TestingNG- Test Method groups (include/exclude) .....	41
Login Incorrect username and password .....	41
TestingNG: @Parameters.....	42
Login in Correct account and password .....	42

Test Suite2 TestingNG- combine LoginTest.....	45
Test Suite 2- Run Login Success and Incorrect Account .....	45
Part 3-2: Creating Test Suite-Driver and Browser .....	49
Test Suite3 TestingNG- Annotations-BeforeMethod and AfterMethod .....	49
Test Suite 3 Add Setup method for Driver .....	49
Test Suite4 Adding Method for browser type .....	54
Part 4: Debugging .....	57
Toggle breakpoint.....	57
Wait Methods.....	59
Test Suite5 Implicit and Explicit wait.....	60
Test Suite6 ElementNotVisibleException .....	61
Test Suite7 Timeout exception.....	65
Test Suite8 No suchElementException .....	67
Test Suite9: Tricky Exception: StaleElementReferenceException .....	72
Part 5: Summary .....	77
Test Suite-Summarize All.....	77
Part 6: Reference .....	78
1.    Update Marven Project .....	78

# Part 1: Environment Setup

## 1. Java and Maven Setup

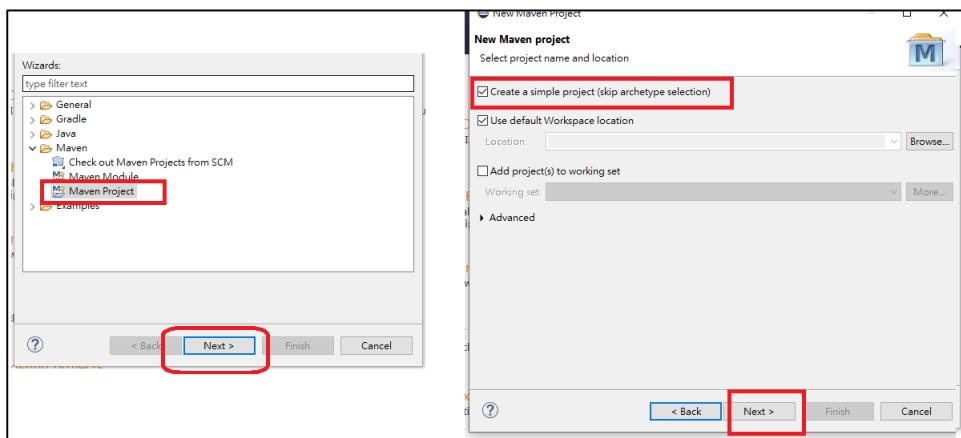
Check your java version

```
java -version // this will check your jre version  
javac -version // this will check your java compiler version if you installed the jdk
```

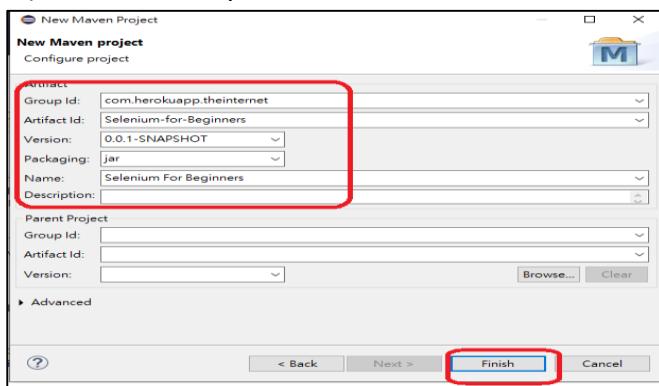
## 2. Eclipse Setting

Create a New Project

- 1.) Create new project (File->new->new project)
- 2.) Select Maven-> Maven Project ->next

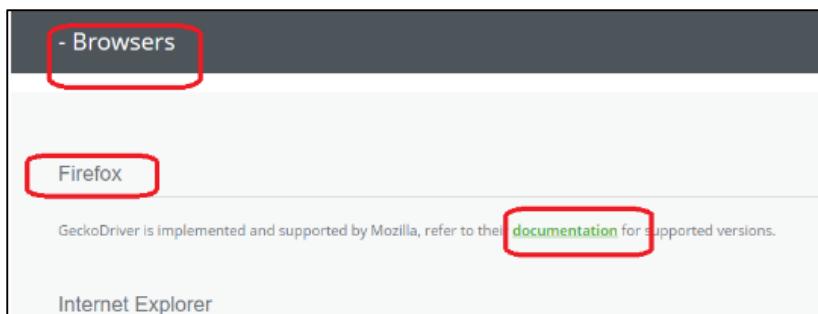


- 3.) Add the Group ID

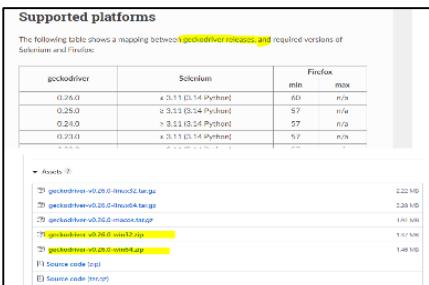


#### 4.) Download Selenium Driver

Go to the official link: <https://selenium.dev/downloads/>, and download the Browser's driver.

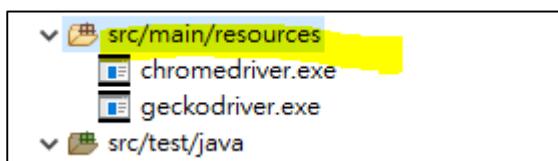


Go to documentation to download chrome and firefox driver.

Firefox:	Chrome:
	
	<p>All versions available in <a href="#">Downloads</a></p> <ul style="list-style-type: none"><li>Latest stable release: <a href="#">ChromeDriver 79.0.3945.36</a></li><li>Previous stable release: <a href="#">ChromeDriver 78.0.3904.105</a></li><li>Latest beta release: ChromeDriver 80 - coming soon</li></ul>

#### 5.) Put the driver in the project folder

You can also drag it into resource folder



#### 6.) Configuring Maven Project

Adding configuration file, plugins and dependencies are Jar files

##### 1. Add Apache Maven Compiler Plugin

Link: <https://maven.apache.org/plugins/maven-compiler-plugin/examples/set-compiler-source-and-target.html>

## 2. Copy the xml text and add into pom.xml

Copy from <built> to </build>

```
<!-->
3. <build>
4. [...]
5. <plugins>
6.   <plugin>
7.     <groupId>org.apache.maven.plugins</groupId>
8.     <artifactId>maven-compiler-plugin</artifactId>
9.     <version>3.8.1</version>
10.    <configuration>
11.      <source>1.8</source>
12.      <target>1.8</target>
13.    </configuration>
14.  </plugin>
15. </plugins>
16. [...]
17. </build>
18. <!-->
```

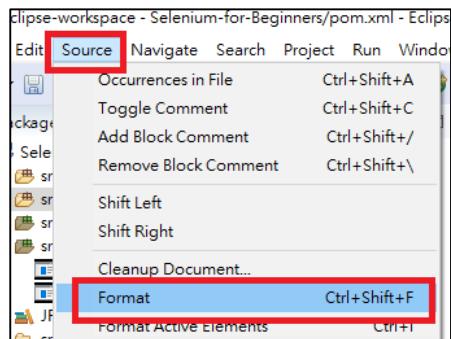
\*Selenium-for-Beginners/pom.xml [x]

```
1<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2<modelVersion>4.0.0</modelVersion>
3<groupId>com.herokuapp.theinternet</groupId>
4<artifactId>Selenium-For-Beginners</artifactId>
5<version>0.0.1-SNAPSHOT</version>
6<name>Selenium For Beginners</name>
7
8
9
10
11
12
13
14</project>
```

→

```
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.1</version>
    <configuration>
      <source>1.8</source>
      <target>1.8</target>
    </configuration>
  </plugin>
</plugins>
</build>
```

Press Ctrl+shift+f or press source->format. to format the code, so look cleaner



## 3. Modify pom.xml into correct java version

I used java 13.0, so I will change to 13. (we can use java --version to see java version)

```
C:\Users\ChenChih>java -version
java version "13.0.1" 2019-10-15
Java(TM) SE Runtime Environment (build 13.0.1+9)
Java HotSpot(TM) 64-Bit Server VM (build 13.0.1+9, mixed mode, sharing)

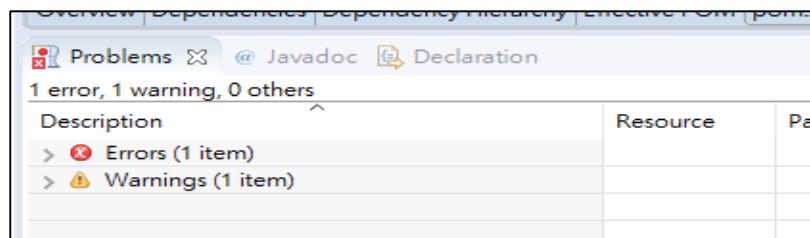
C:\Users\ChenChih>javac -version
javac 13.0.1
```

Modify as:

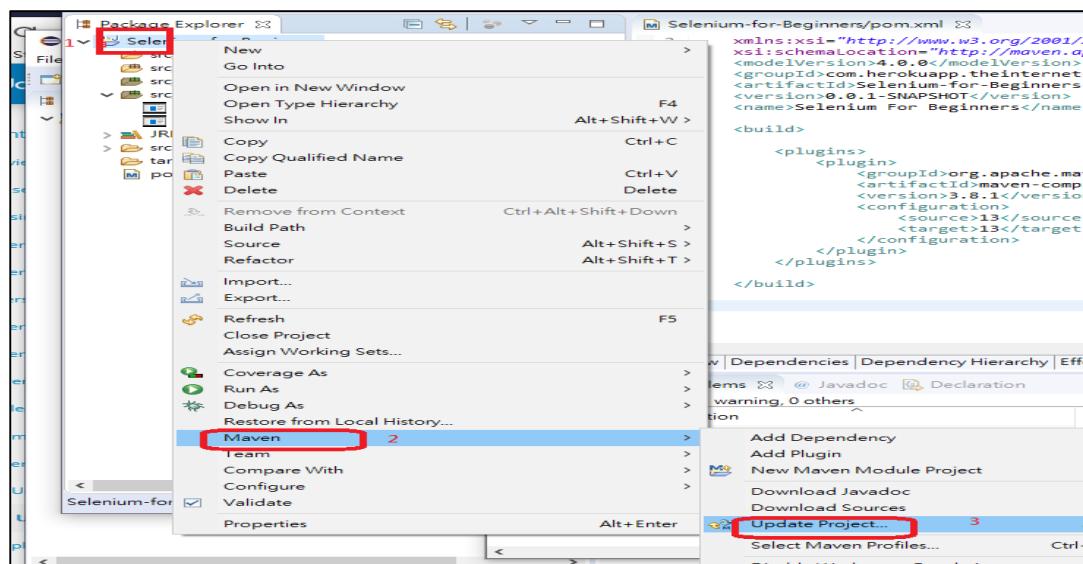
```
<!-->
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.1</version>
    <configuration>
      <source>13</source>
      <target>13</target>
    </configuration>
  </plugin>
</plugins>
```

#### 4. Make sure problem will not have warning or error

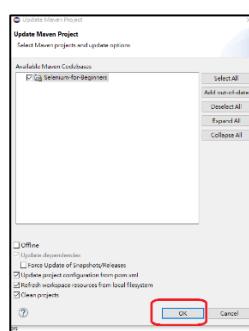
After save it button will occur error



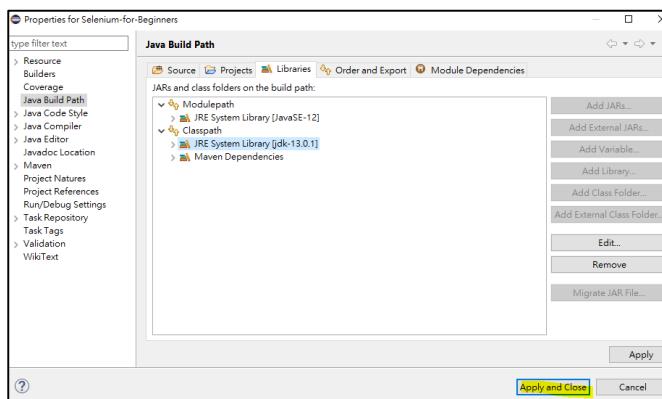
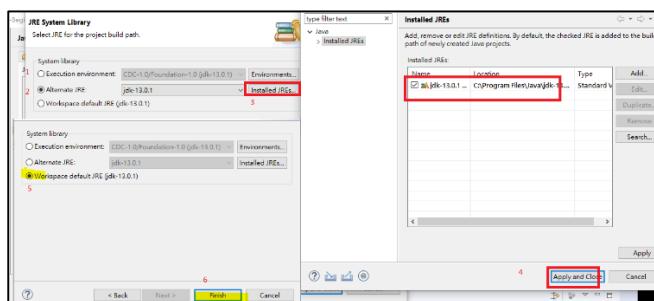
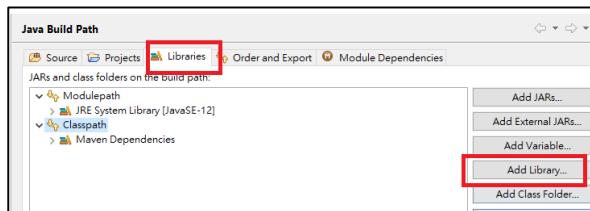
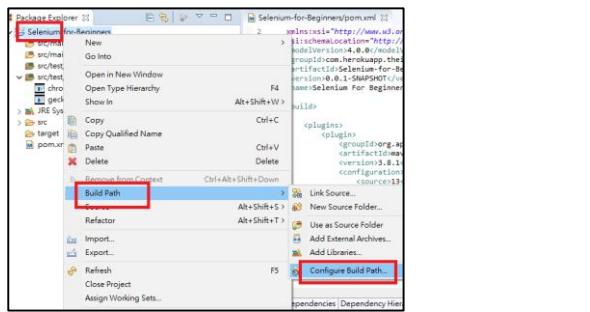
Fixed by using this: right click and press Maven>update project



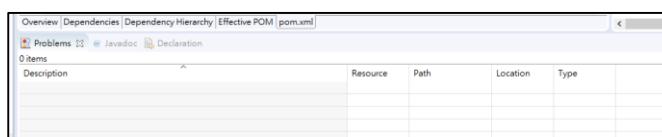
Press ok



5. If {problem still occur warning, please do below step. (if no warning you can skip). Your JDK might not match your Eclipse, need to do some setting}



And go to update maven project again (just like you did in step 9). No error or warning as below



## Maven Surefire Plugin

### 1.) Create Maven Surefire Plugin file

#### 1. Go to Maven Surefire link as below

<http://maven.apache.org/surefire/maven-surefire-plugin/>

#### Examples

The following examples show how to use the Surefire Plugin in more advanced use cases:

- Using TestNG
- Using JUnit 5 Platform
- Using JUnit
- Using POJO Tests

#### 2. Copy the XML code to POM.xml(java package)

Press “Using Testing” from above link, and copy “Using Suite XML Files ‘s plugin” and paste into pom.xml

### Using Suite XML Files

Another alternative is to use TestNG suite XML files. This allows flexible configuration of the tests to be run. The Surefire Plugin configuration:

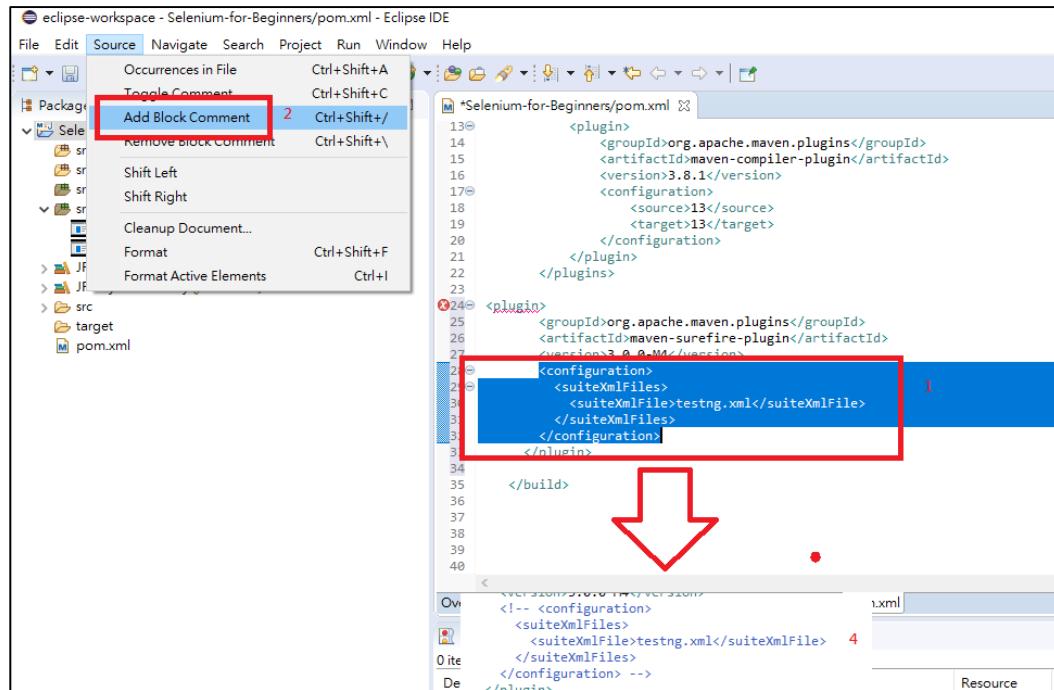
```
1. <plugins>
2.   [...]
3.   <plugin>
4.     <groupId>org.apache.maven.plugins</groupId>
5.     <artifactId>maven-surefire-plugin</artifactId>
6.     <version>3.0.0-M4</version>
7.     <configuration>
8.       <suiteXmlFiles>
9.         <suiteXmlFile>testng.xml</suiteXmlFile>
10.      </suiteXmlFiles>
11.    </configuration>
12.  </plugin>
13. [...]
14. </plugins>
```

Copy to pom.xml as below

```
22   </plugins>
23
24<plugin>
25   <groupId>org.apache.maven.plugins</groupId>
26   <artifactId>maven-surefire-plugin</artifactId>
27   <version>3.0.0-M4</version>
28   <configuration>
29     <suiteXmlFiles>
30       <suiteXmlFile>testng.xml</suiteXmlFile>
31     </suiteXmlFiles>
32   </configuration>
33 </plugin>
34
35 </build>
36
37
```

### 3. Comment configuration (we don't need configuration, so mark as comment).

Note: Current test we still don't need to used configuration



Press format to make your code nicer. Update maven project again

## 2.) Set Testing dependency (6.14.3) {Testing framework for Java}

### 1. Copy TestNG code to pom.xml

<https://mvnrepository.com/artifact/org.testng/testng>

Copy below

The screenshot shows a search interface for Maven dependencies. At the top, there are tabs for "Maven", "Gradle", "SBT", "Ivy", "Grape", "Leiningen", and "Buildr". Below the tabs, the URL "https://mvnrepository.com/artifact/org.testng/testng" is entered. The search results show a single dependency entry for org.testng:testng:6.14.3. The dependency code is displayed in a code editor-like area:

```
<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>6.14.3</version>
    <scope>test</scope>
</dependency>
```

At the bottom of the code editor, there is a checkbox labeled "Include comment with link to declaration" which is checked.

2. Add <dependencies> tag as below paste it here and save.

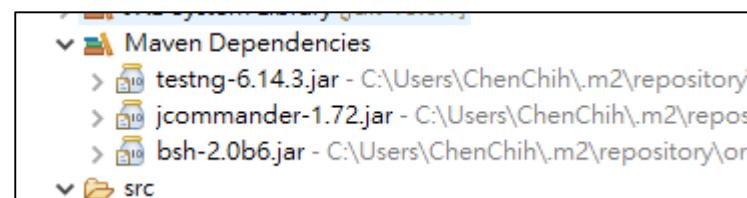
```
<artifactId>Selenium-Tor-Beginners</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>Selenium For Beginners</name>

<dependencies>
    <!-- https://mvnrepository.com/artifact/org.testng/testng -->
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>6.14.3</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>

```

3. Update Maven Project will show below



3.) Set Selenium java dependency (3.141.5){ Selenium Java}

1. Copy Selenium code to pom.xml as below link

<https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>

2. Copy below text

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
</dependency>
```

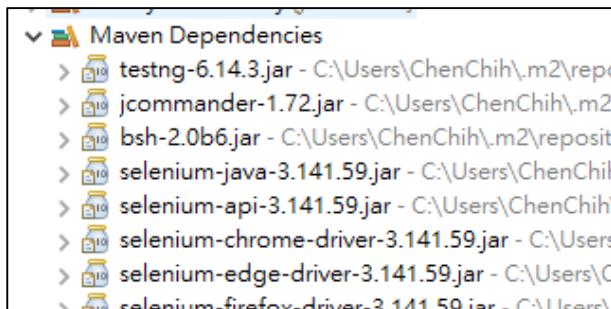
3. Copy to pom.xml and save

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
</dependency>
```

#### 4. Update maven project

Go to reference for setting update marven if you don't know.

After update Marven Project, Maven Dependencies will have many package as below:



#### 4.) Summary of all Plugin or tag file

```
9
10<dependencies>
11    <!-- https://mvnrepository.com/artifact/org.testng/testng -->
12    <dependency>
13        <groupId>org.testng</groupId>
14        <artifactId>testng</artifactId>
15        <version>6.14.3</version>
16    </dependency>
17    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/java -->
18    <dependency>
19        <groupId>org.seleniumhq.selenium</groupId>
20        <artifactId>selenium-java</artifactId>
21        <version>3.141.59</version>
22    </dependency>
23</dependencies>
24
25
26<build>
27
28    <plugins>
29        <plugin>
30            <groupId>org.apache.maven.plugins</groupId>
31            <artifactId>maven-compiler-plugin</artifactId>
32            <version>3.8.1</version>
33            <configuration>
34                <source>13</source>
35                <target>13</target>
36            </configuration>
37        </plugin>
38    </plugins>
39</build>
40
41<plugin>
42    <groupId>org.apache.maven.plugins</groupId>
43    <artifactId>maven-surefire-plugin</artifactId>
44    <version>3.0.0-M4</version>
45    <!-- <configuration> <suiteXmlFiles> <suiteXmlFile>testng.xml</suiteXmlFile>
46        </configuration> </suiteXmlFiles> </configuration> -->
47</plugin>
48</plugins>
```

The code block shows a portion of a Maven POM XML file. Several sections are highlighted with red boxes and labeled:

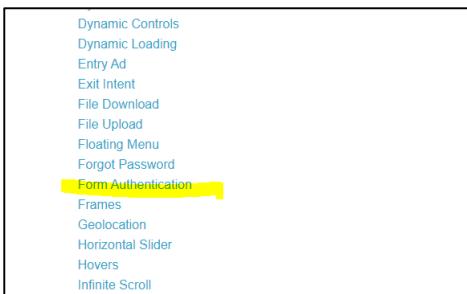
- A red box highlights the first dependency section (Testng dependency).
- A red box highlights the second dependency section (Selenium java).
- A red box highlights the configuration of the maven-compiler-plugin (Apache Maven Compiler).
- A red box highlights the configuration of the maven-surefire-plugin (Maven Surefire Plugin).

# Part 2: Create Test Plan

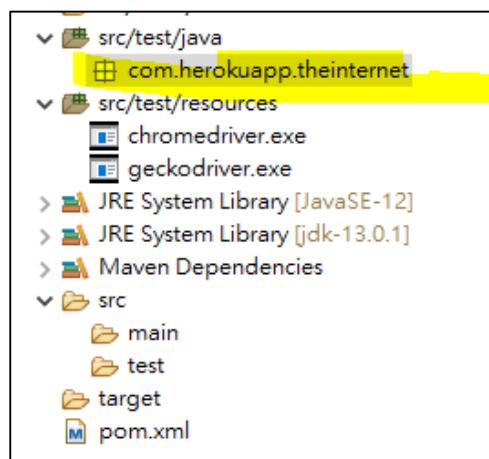
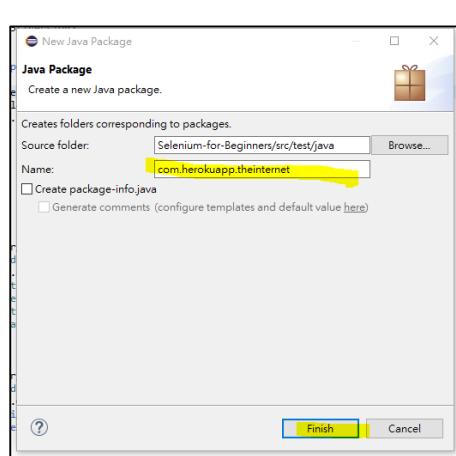
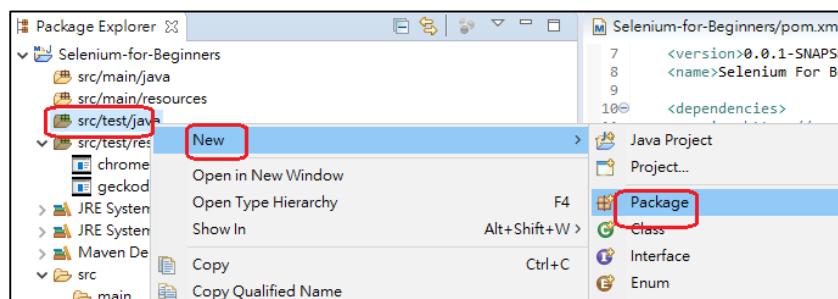
## Test Plan1-Create Open Web Browser

1. Create Test Case, used which type of link you want to test

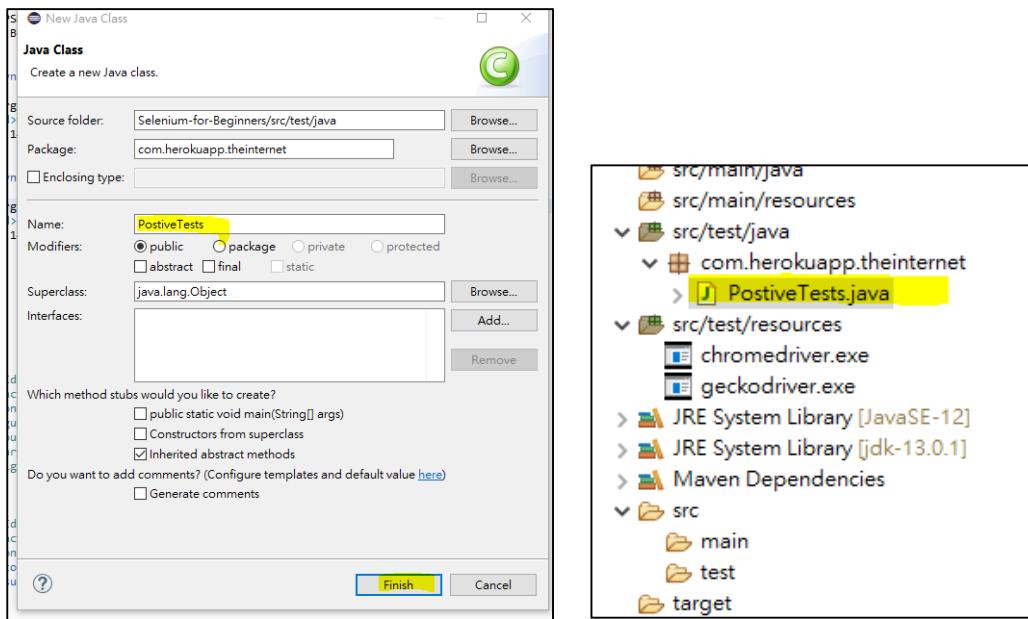
Test URL: <http://the-internet.herokuapp.com/>



2. Create package



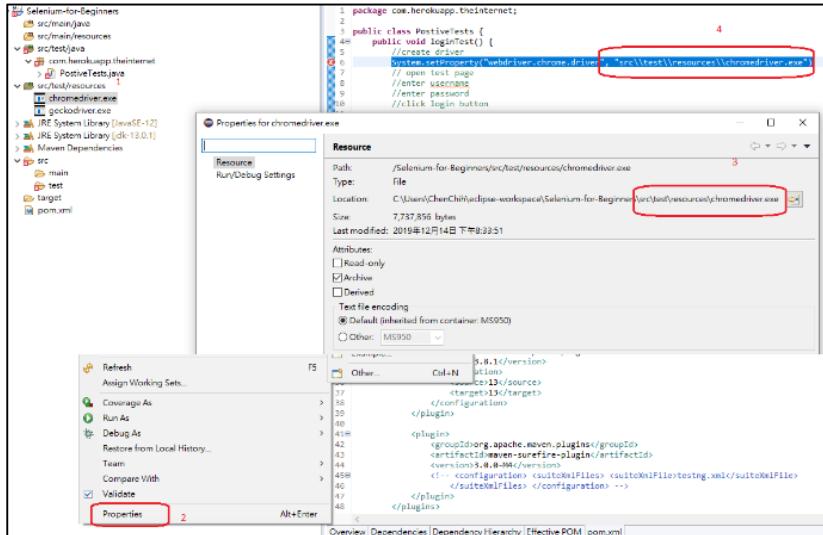
### 3. Create class name PsostiveTest.java



### 4. Import driver in the file PositiveTets.java

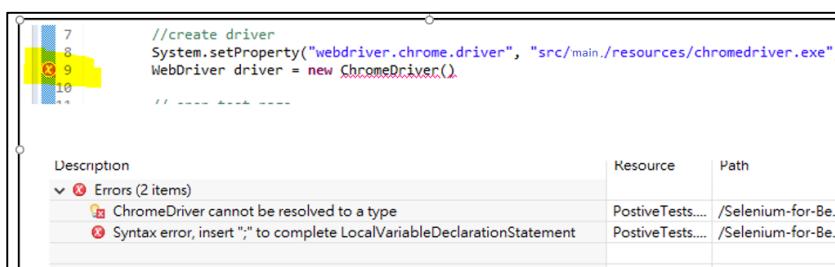
Syntax: `System.setProperty(key, value)`

Find your Driver store location, by properties and copy the location, as show below.



`System.setProperty("webdriver.chrome.driver",`

`src/main/resources/chromedriver.exe")`



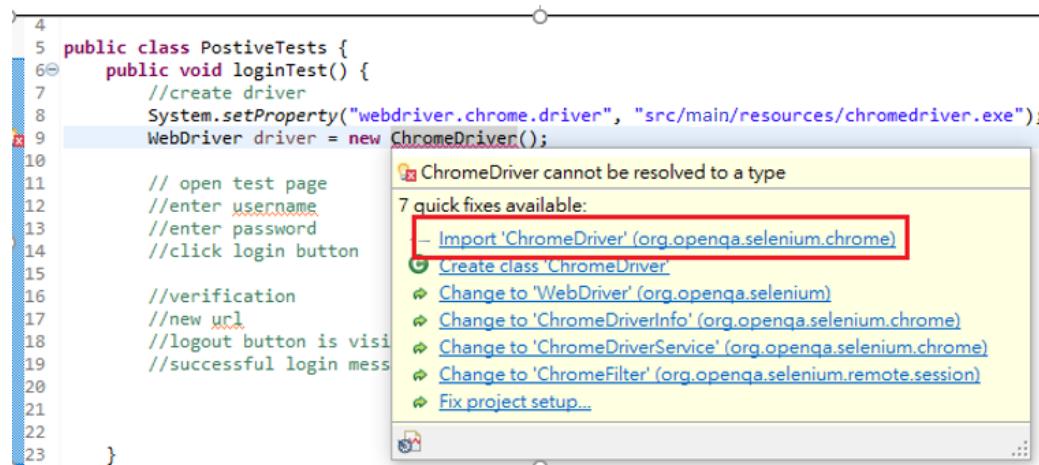
Will occur error, due to not import driver.

Import selenium driver and chrome driver:

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

```
1 package com.herokuapp.theinternet;
2
3 import org.openqa.selenium.WebDriver;
4
5 public class PositiveTests {
6     public void loginTest() {
7         //create driver
8         System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
9         WebDriver driver = new ChromeDriver();
10
11         // open test page
12         //enter username
13         //enter password
14         //click login button
15
16         //verification
17         //new url
18         //logout button is visible
19         //successful login message
20     }
}
```

We can use Eclipse method to import it:



The screenshot shows the Eclipse IDE interface with Java code in the editor. A tooltip is displayed over the word 'ChromeDriver' at line 9, which is highlighted in blue. The tooltip contains the message 'ChromeDriver cannot be resolved to a type' and '7 quick fixes available'. One fix, 'Import 'ChromeDriver' (org.openqa.selenium.chrome)', is highlighted with a red border.

```
4
5 public class PositiveTests {
6     public void loginTest() {
7         //create driver
8         System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
9         WebDriver driver = new ChromeDriver();
10
11         // open test page
12         //enter username
13         //enter password
14         //click login button
15
16         //verification
17         //new url
18         //logout button is visible
19         //successful login mess
20     }
}
```

Update maven project

5. create driver and open webpage

```
String url ="http://the-internet.herokuapp.com/login";
driver.get(url);
```

6. Adding Max Screen

```
// maximize browser window
driver.manage().window().maximize();
```

7. Sleep Function to sleep second

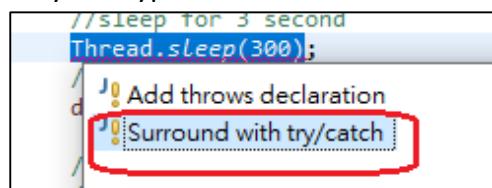
sleep 3 second

```
Thread.sleep(300);
```

## 8. Extract sleep method

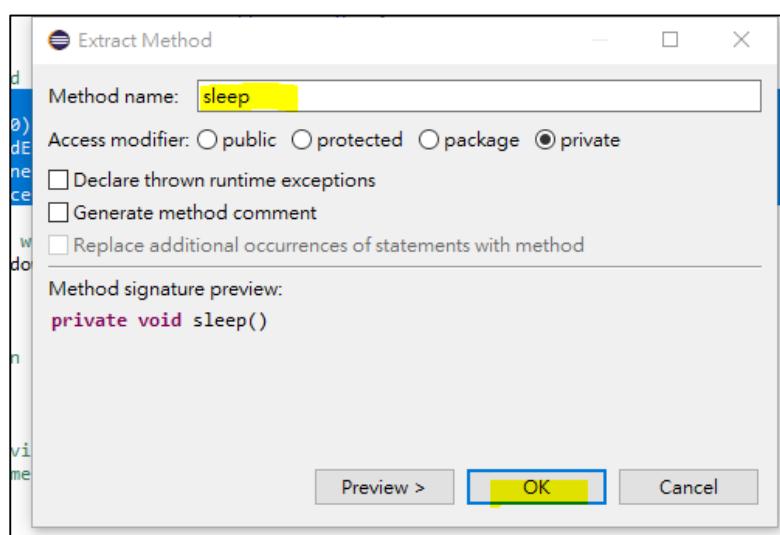
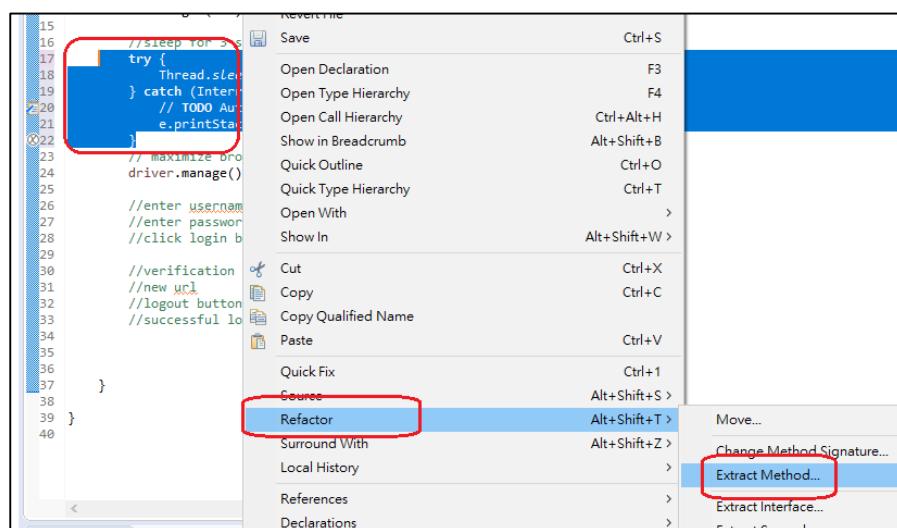
Adding sleep method from previous for better coding

The purpose of this will let code to be more efficient, used don't have to type 3000, they can type n second.



```
//sleep for 3 second
try {
    Thread.sleep(300);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
```

Extract the code:



Sleep will create a function as below:

```
7@ public void loginTest() {
8    //create driver
9    System.setProperty("webdriver.chrome.driver", "src/test/resources/
10    WebDriver driver = new ChromeDriver();
11
12    // open test page
13    String url ="http://the-internet.herokuapp.com/login";
14    driver.get(url);
15
16    //sleep for 3 second
17    sleep();
18
19    // maximize browser window
20    driver.manage().window().maximize();
21
22    //enter username
23    //enter password
24    //click login button
25
26    //verification
27    //new url
28    //logout button is visible
29    //successful login message
30
31
32 }
33
34
35@ private void sleep() {
36     try {
37         Thread.sleep(300);
38     } catch (InterruptedException e) {
39         // TODO Auto-generated catch block
40         e.printStackTrace();
41     }
42 }
43 }
```

## 9. Modify sleep()

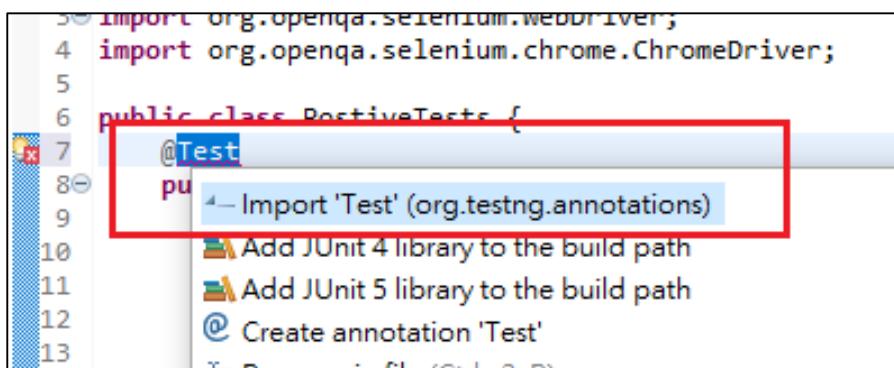
move sleep() under driver, and edit the function. Add variable to it

```
39
40@ private void sleep(long m) {
41     try {
42         Thread.sleep(m);
43     } catch (InterruptedException e) {
44         // TODO Auto-generated catch block
45         e.printStackTrace();
46     }
47 }
48
49 }
```

```
5
6 public class PositiveTests {
7@     public void loginTest() {
8        //create driver
9        System.setProperty("webdriver.chrome.driver", "src/test/resources/chromedriver");
10       WebDriver driver = new ChromeDriver();
11
12       System.out.println("Starting login page ");
13
14       //sleep for 3 second
15       sleep(3000);
16
17       // maximize browser window
18       driver.manage().window().maximize();
19
20       // open test page
21       String url ="http://the-internet.herokuapp.com/login";
22       driver.get(url);
23       System.out.println("Page is opened");
24
25       //sleep for 2 second
26       sleep(2000);
27
28
29       //enter username
30       //enter password
31       //click login button
32
33       //verification
34       //new url
35       //logout button is visible
36       //successful login message
37
38
39 }
```

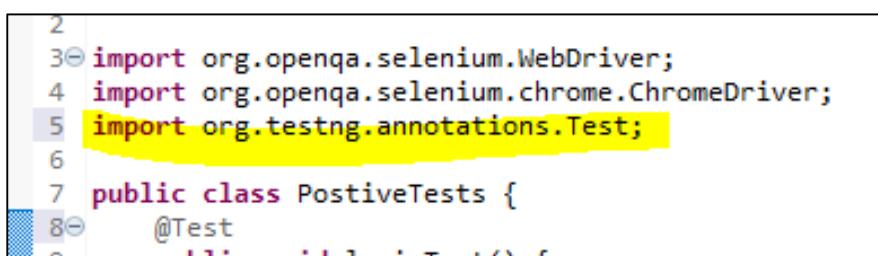
## 10. Add @test notation

The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.



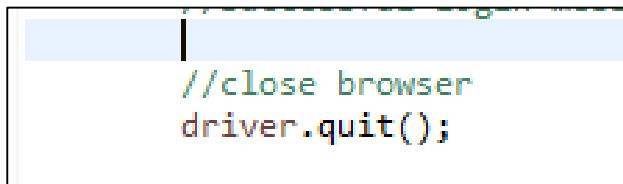
A screenshot of an IDE showing Java code. Line 7 contains the annotation `@Test`. A red box highlights the code completion dropdown that appears when the cursor is over the annotation. The dropdown shows the following options:

- Import 'Test' (org.testng.annotations)
- Add JUnit 4 library to the build path
- Add JUnit 5 library to the build path
- Create annotation 'Test'



A screenshot of an IDE showing Java code. Line 5 now contains the import statement `import org.testng.annotations.Test;`, which is highlighted with a yellow background. The rest of the code remains the same as in the previous screenshot.

## 11. Close Browser



A screenshot of an IDE showing Java code. The code includes the line `driver.quit();` which is used to close the browser after the test has run.

## Test Plan2- Run Open Web Browser

Description: This Test is just open browser

File used:

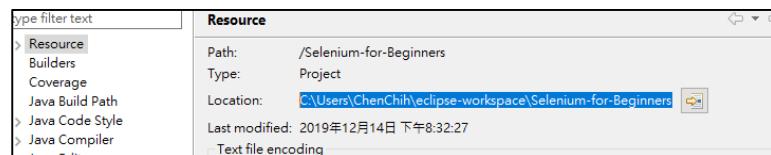
PositiveTests.java

pom.xml

Folder name: Case1-1\_openWEB

### 1. Check your file directory and location

Press right click>properties to find your package location.



### 2. Go to CMD prompt to run test

Execute: mvn clean test

```
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.herokuapp.theinternet.PositiveTests
Starting ChromeDriver 79.0.3945.16 (93fcc21110c10dbbd49bfff8f472335360e31d05-refs/branch-heads/0
Only local connections are allowed.
Please protect ports used by ChromeDriver and related test frameworks to prevent access
[1576374537,597][WARNING]: Timed out connecting to Chrome, retrying...
12月 15, 2019 9:48:59 上午 org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Starting login page
[1576374544,650][WARNING]: Timed out connecting to Chrome, retrying...
Page is opened
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 22.084 s - in co
sts
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 45.674 s
[INFO] Finished at: 2019-12-15T09:49:13+08:00
[INFO] -----
```

C:\Users\ChenChih\eclipse-workspace\Selenium-for-Beginners>mvn clean test

### If fail, might be browser driver not match your browser's version.

### 3. Add test case under prompt

mvn -Dtest=PositiveTests clean test

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 21.488 s - in com.herokuapp
sts
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 24.252 s
[INFO] Finished at: 2019-12-15T10:11:11+08:00
[INFO] -----
```

C:\Users\ChenChih\eclipse-workspace\Selenium-for-Beginners>mvn -Dtest=PositiveTests clean test

## 4. Use Eclipse environment to run mvn

Set up MVN Environment configuration first

The screenshots illustrate the process of setting up a Maven build configuration in Eclipse:

- Screenshot 1: Eclipse Context Menu**  
Shows the context menu for a Java file named `PositiveTest.java`. The "Run As" option is highlighted, and a sub-menu "Run Configurations..." is also highlighted.
- Screenshot 2: Run Configurations Dialog**  
Shows the "Run Configurations" dialog. A "New Configuration" prototype is selected. The "Goals" field contains `-Dtest=PositiveTests clean test`. The "Base directory" dropdown is set to "Workspace...".
- Screenshot 3: Choose Base Directory Dialog**  
Shows the "Choose Base directory" dialog with the path `Selenium-for-Beginners` selected.
- Screenshot 4: Run Configurations Dialog (Final State)**  
Shows the "Run Configurations" dialog with the configuration name "PositiveTests" and the same settings as the previous step. The "Run" button is visible at the bottom.
- Screenshot 5: Console Output**  
Shows the Eclipse Console output window with the following log entries:  

```
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 21.602 s
[INFO] Finished at: 2019-12-15T10:30:29+08:00
[INFO] -----
```

# Understand Locator

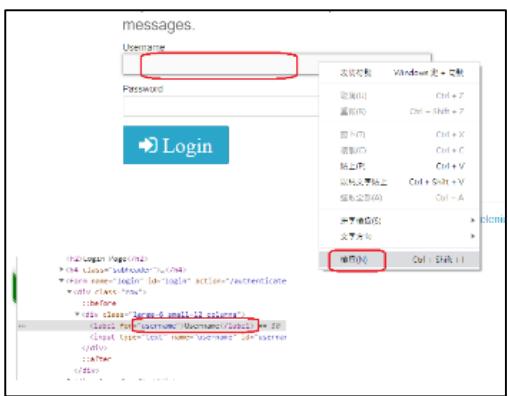
## Locator tutorial:

<https://www.guru99.com/locators-in-selenium-ide.html>

[https://www.protechtraining.com/content/selenium\\_tutorial-locators](https://www.protechtraining.com/content/selenium_tutorial-locators)

You can also use “Ranorex Selocity” plugin in chrome, it will automatically help us select the id.

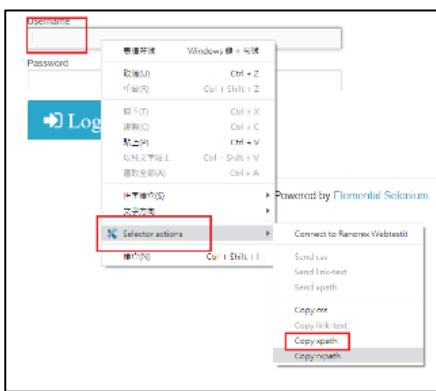
Normal condition with chrome inspect



## With Ranorex Selocity plugin



It will automatically copy the id for you

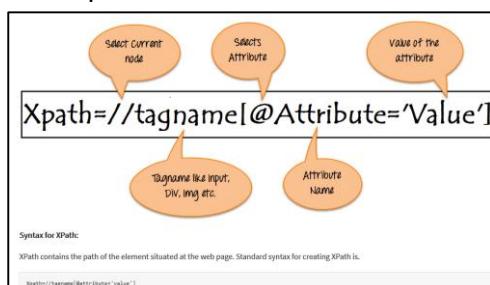


## 1. Xpath

Xpath: //tag[@attrribue='value']

Example: Xpath: /html//input[@id='username']

Css: input#username



use chrome console to check

```
> $x("//html//input[@id='username'])  
  < [input#username]
```

## Test Plan3 Create login user and password page

Description: Enter user and password

### File used:

PositiveTests.java

pom.xml

Folder name: Case1-2\_enter\_username\_password

### 1. Create webElement for username and password

```
//enter username  
WebElement username =driver.findElement(By.id("username"));  
//enter password  
WebElement password =driver.findElement(By.name("password"));  
//click login button  
WebElement logInButton =driver.findElement(By.tagName("button"));
```

```

4. import org.openqa.selenium.WebDriver;
5. import org.openqa.selenium.WebElement;
6. import org.openqa.selenium.chrome.ChromeDriver;
7. import org.testng.annotations.Test;
8.
9. public class PositiveTests {
10.     @Test
11.     public void loginTest() {
12.         //create driver
13.         System.setProperty("webdriver.chrome.driver", "src/test/resources/chromedriver.exe");
14.         WebDriver driver = new ChromeDriver();
15.
16.         System.out.println("Starting login page ");
17.
18.         //sleep for 3 second
19.         sleep(3000);
20.
21.         // maximize browser window
22.         driver.manage().window().maximize();
23.
24.         // open test page
25.         String url = "http://the-internet.herokuapp.com/login";
26.         driver.get(url);
27.         System.out.println("Page is opened");
28.
29.         //sleep for 2 second
30.         sleep(2000);
31.
32.
33.         //enter username
34.         WebElement username = driver.findElement(By.id("username"));
35.
36.         //enter password
37.         WebElement password = driver.findElement(By.name("password"));
38.
39.         //click login button
40.         WebElement loginButton = driver.findElement(By.tagName("button"));
41.
42.         //verification
43.         //new url
44.         //logout button is visible

```

## 2. Logout button (after login)

Find xpath value

Used console to find the xpath value

Go to inspect

```

You logged into a secure area!
Secure Area
Welcome to the Secure Area. When you are done click lo
Logout
click logout below.</h4>
<a class="button secondary radius" href="/logout">
  <i class="icon-2x icon-signout"> Logout</i>
</a>
</div>

```

Go under console and enter below to make sure only one button

`$x("//a[@class='button secondary radius'])")`

```

> $x("//a[@class='button secondary radius']")
< ▾ [a.button.secondary.radius] ⓘ
  ▶ 0: a.button.secondary.radius
    length: 1
  ▶ __proto__: Array(0)
>

```

WebElement `logOutButton =driver.findElement(By.xpath("//a[@class='button secondary radius']"));`

```

//logout button is visible
WebElement logOutButton =driver.findElement(By.xpath("//a[@class='button secondary radius']"));

```

### 3. successful login message

The screenshot shows a web page titled "Secure Area". A green header bar at the top contains the message "You logged into a secure area!". Below this, there is a "Logout" button and a note saying "Welcome to the Secure Area. When you are done click logout below.". At the bottom, it says "Powered by Elemental Selenium". To the right of the page, the browser's developer tools are open, specifically the "Elements" tab. It shows the HTML structure of the page, including the message box and its CSS classes.

Use selector actions to copy css's div ("flash")

The screenshot shows a context menu in a browser window. The menu has several options like "Copy CSS", "Copy Attributes", "Copy XPath", and "Copy iPath". The "Copy CSS" option is highlighted with a red box. The background shows a "Secure Area" page with a "Logout" button.

Or inspect

The screenshot shows the browser's developer tools with the "Elements" tab selected. It displays the HTML code for the success message. The code includes a div with id="flash" and class="flash success", containing the text "You logged into a secure area!" and a close link.

```
WebElement successMessage =driver.findElement(By.cssSelector("#flash"));
```

```
//successful login message  
WebElement successMessage =driver.findElement(By.cssSelector("#flash"));  
//close browser  
driver.quit();|
```

#### 4. Type in correct username and password

```
//enter username
WebElement username =driver.findElement(By.id("username"));
username.sendKeys("tomsmith");
sleep(1000);
//enter password
WebElement password =driver.findElement(By.name("password"));
password.sendKeys("SuperSecretPassword!");
sleep(3000);
//click login button
WebElement logInButton =driver.findElement(By.tagName("button"));
logInButton.click();
sleep(5000);

//verification
//new url
//logout button is visible
WebElement logOutButton =driver.findElement(By.xpath("//a[@class='button secondary radius']"));

//successful login message
WebElement successMessage =driver.findElement(By.cssSelector("#flash"));
//close browser
driver.quit();
```

#### 5. Create Assertion

Assertion is use to compare actual and expected value is correct or not

##### Verification1- create assertion compare actual and expected

```
1 import org.openqa.selenium.WebDriver;
2 import org.testng.Assert;
3 import org.testng.annotations.Test;
4

//verification
String expectedUrl= "http://the-internet.herokuapp.com/secure";
String actualUrl=driver.getCurrentUrl();
Assert.assertEquals(actualUrl, expectedUrl, "Actual page url is not the same as expected");

//new url
```

```
<terminated> PositiveTests [maven-build] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (2019-12-20T21:26:12+08:00)
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  35.474 s
[INFO] Finished at: 2019-12-20T21:26:12+08:00
[INFO] -----
```

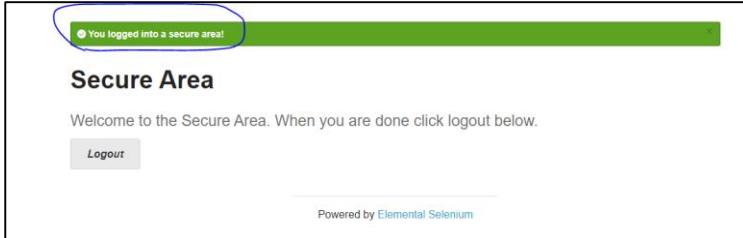
##### Verification2-verifiy logout button using Assert.True

```
//new url
//logout button is visible
WebElement logOutButton =driver.findElement(By.xpath("//a[@class='button secondary radius']"));
Assert.assertTrue(logOutButton.isDisplayed(), "Log out button is not visible");
```

### Verification3 message (will be fail)

```
3
4      //new url
5      //logout button is visible
6      WebElement logOutButton =driver.findElement(By.xpath("//a[@class='button secondary radius']"));
7      Assert.assertTrue(logOutButton.isDisplayed(), "Log out button is not visible" );
8
9      //successful login message
10     WebElement successMessage =driver.findElement(By.cssSelector("#flash"));
11     String expectedMessage= "You logged into a secure area!";
12     String actualMessage =successMessage.getText();
13     Assert.assertEquals(actualMessage, expectedMessage, "Actual Message is not the same as expected");
14
15 }
```

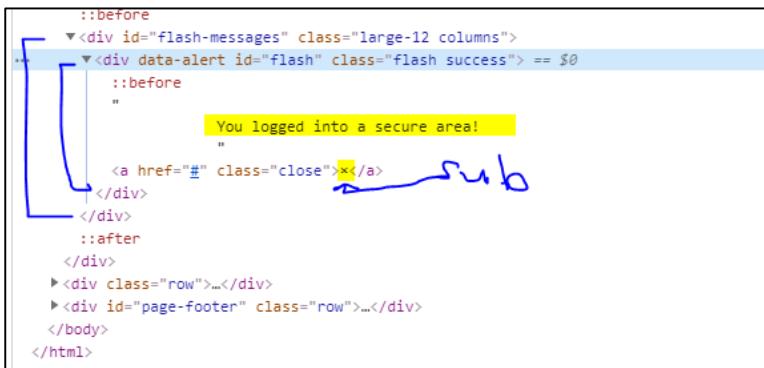
Expected Message is:



### Why Fail?

```
Page is opened
[ERROR] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 28.448 s <<< FAILURE! - in com.herokuapp.theinternet.PositiveTests
[ERROR] com.herokuapp.theinternet.PositiveTests.loginTest  Time elapsed: 28.091 s  <<< FAILURE!
java.lang.AssertionError:
  Actual Message is not the same as expected expected [You logged into a secure area!] but found [You logged into a secure area!
x]
```

Due to compare fail, due to the message have a X. please inspect the page for more information as below:



Look at the div element, it also contains a sub div element, X

### Verification3 message (will be pass, correct mistake)

Using Assert.assertTrue to solve this issue

```
// successful login message
WebElement successMessage = driver.findElement(By.cssSelector("#flash"));
String expectedMessage = "You logged into a secure area!";
String actualMessage = successMessage.getText();
// Assert.assertEquals(actualMessage, expectedMessage, "Actual Message is not
// the same as expected");
Assert.assertTrue(actualMessage.contains(expectedMessage));
    "Actual Message does not contain expected message.\nActual Message: " + actualMessage
    + "\nExpectedMessage: " + expectedMessage);

// close browser
```

If fail will output like this, I try to change expected to --

The screenshot shows an IDE interface with a code editor and a terminal window. The code editor contains Java code for a login test. The terminal window shows the execution of the test and the resulting failure message.

```
45     // verification
46     String expectedUrl = "http://the-internet.herokuapp.com/secure";
47     String actualUrl = driver.getCurrentUrl();
48     Assert.assertEquals(actualUrl, expectedUrl, "Actual page url is not the same as expected");
49
50     // new url
51     // logout button is visible
52     WebElement logOutButton = driver.findElement(By.xpath("//a[@class='button secondary radius']"));
53     Assert.assertTrue(logOutButton.isDisplayed(), "Log out button is not visible");
54
55     // successful login message
56     WebElement successMessage = driver.findElement(By.cssSelector("#flash"));
57     String expectedMessage = "You logged into a secure area--!";
58     String actualMessage = successMessage.getText();
59     // Assert.assertEquals(actualMessage, expectedMessage, "Actual Message is not
60     // the same as expected");
61     Assert.assertTrue(actualMessage.contains(expectedMessage),
62         "Actual Message does not contain expected message.\nActual Message: " + actualMessage
63         + "\nExpectedMessage: " + expectedMessage);
64
65
```

TERMINATED PostiveTests [Maven Build] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (2019年12月20日下午10:03:13)

[ERROR] Failures:

[ERROR] PostiveTests.loginTest:62 Actual Message does not contain expected message.  
Actual Message: You logged into a secure area!

ExpectedMessage: You logged into a secure area--! expected [true] but found [false]

[INFO]

[ERROR] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0

[INFO]

[INFO] -----

[INFO] BUILD FAILURE

## 6. Summary Assert

`assertEquals(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

assertEquals will print.

`assertTrue(boolean condition, String message) : void - Assert`

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

assertTrue will not print message, so we need to type in the message, but  
So since we want assert to be true, but it returns false.

# Test Plan4 Incorrect username test

Description: Enter incorrect user and password

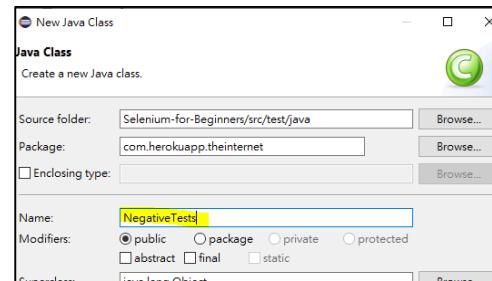
File used:

PositiveTests.java

pom.xml

Folder name: Case1-3-assertion

## 1. Create new class named NegativeTests



Add `public void incorrectUsernameTest()`

## 2. Copy the code of positive

from creating diver ~ login button

```
1 public class NegativeTests {  
2     @Test  
3     public void incorrectUsernameTest() {  
4         // create driver  
5         System.setProperty("webdriver.chrome.driver", "src/test/resources/chromedriver.exe");  
6         WebDriver driver = new ChromeDriver();  
7  
8         System.out.println("Starting login page ");  
9  
10        // sleep for 3 second  
11        sleep(3000);  
12  
13        // maximize browser window  
14        driver.manage().window().maximize();  
15  
16        // open test page  
17        String url = "http://the-internet.herokuapp.com/login";  
18        driver.get(url);  
19        System.out.println("Page is opened");  
20  
21        // sleep for 2 second  
22        sleep(2000);  
23  
24        // enter username  
25        WebElement username = driver.findElement(By.id("username"));  
26        username.sendKeys("tomsmith");  
27        sleep(1000);  
28        // enter password  
29        WebElement password = driver.findElement(By.name("password"));  
30        password.sendKeys("SuperSecretPassword!");  
31        sleep(3000);  
32        // click login button  
33        WebElement loginButton = driver.findElement(By.tagName("button"));  
34        loginButton.click();  
35        sleep(5000);  
36    }  
37}
```

Also copy sleep function in the bottom

## 3. Changed some value in the code

*changed the username to wrong username or used below*

```
username.sendKeys("incorrectUsername");
```

*and also changed*

```
System.out.println("Starting incorrectUsername Test ");
```

#### 4. Add verification

check error message if type in and inspect it



We will use id

```
<div id="flash-messages" class="large-12 columns">
  <div data-alert id="flash" class="flash error">= $0
    ::before
    "
      Your username is invalid!
    "
    <a href="#" class="close">×</a>
  </div>
  ::after
</div>
```

```
sleep(5000);
//verification
WebElement ErrorMessage = driver.findElement(By.id("flash"));
String expectedErrorMessage = "Your username is invalid!";
String actualErrorMessage=ErrorMessage.getText();
Assert.assertTrue(actualErrorMessage.contains(expectedErrorMessage),
    "Actual Error Message does not contain expected.\nActual: " + actualErrorMessage
    + "\nExpected: " + expectedErrorMessage);
//close browser
driver.quit();
}
```

#### 5. Run Test

```
sleep(5000);
//verification
WebElement ErrorMessage = driver.findElement(By.id("flash"));
String expectedErrorMessage = "Your username is invalid!";
String actualErrorMessage=ErrorMessage.getText();
Assert.assertTrue(actualErrorMessage.contains(expectedErrorMessage),
    "Actual Error Message does not contain expected.\nActual: " + actualErrorMessage
    + "\nExpected: " + expectedErrorMessage);
//close browser
driver.quit();
}
```

#### 6. Break it, by enter correct username

change `username.sendKeys("tomsmith");`

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

```
java.lang.AssertionError: Actual Error Message does not contain expected.
  Actual: You logged into a secure area!
  ◆ nExpected: Your username is invalid! expected [true] but found [false]
  └ at org.testng.Assert.fail(Assert.java:96)
```

## Test Plan5 Incorrect password test

Description: Enter incorrect user and password

File used:

PositiveTests.java

pom.xml

Folder name: Case1-3-assertion

1. Copy previous code, and create new function and name incorrectPasswordTest

```
public void incorrectPasswordTest()
```

2. change password to incorrect

```
username.sendKeys("tomsmith");  
password.sendKeys("incorrectPassword!");
```

3. Copy error message when enter wrong password

-Got to the web

-login correct user, and wrong password

-Error message as below, copy the message and copy to the code



4. copy error message to the code

```
String expectedErrorMessage = "Your password is invalid!";
```

```

public void incorrectPasswordTest() {
    System.out.println("Starting incorrect Password Test ");
    // create driver
    System.setProperty("webdriver.chrome.driver", "src/test/resources/chromedriver.exe");
    WebDriver driver = new ChromeDriver();

    // sleep for 3 second
    sleep(3000);

    // maximize browser window
    driver.manage().window().maximize();

    // open test page
    String url = "http://the-internet.herokuapp.com/login";
    driver.get(url);
    System.out.println("Page is opened");

    // sleep for 2 second
    sleep(2000);

    // enter username
    WebElement username = driver.findElement(By.id("username"));
    username.sendKeys("tomsmith");
    sleep(1000);
    // enter password
    WebElement password = driver.findElement(By.name("password"));
    password.sendKeys("incorrectPassword!");
    sleep(3000);
    // click login button
    WebElement logInButton = driver.findElement(By.tagName("button"));
    logInButton.click();
    sleep(5000);
    // verification
    WebElement ErrorMessage = driver.findElement(By.id("flash"));
    String expectedErrorMessage = "Your password is invalid!";
    String actualErrorMessage=ErrorMessage.getText();
    Assert.assertTrue(actualErrorMessage.contains(expectedErrorMessage),
        "Actual Error Message does not contain expected.\nActual: " + actualErrorMessage
        + "\nExpected: " + expectedErrorMessage);
}

```

5. comment @test for negative test else it will also run it

```

public class NegativeTests {
    // @Test
    public void incorrectUsernameTest() {
        System.out.println("Starting incorrectUsername Test ");
    }
}

```

6. you can see the function

```

com.herokuapp.theinternet
  NegativeTests
    ● incorrectUsernameTest():void
    ● incorrectPasswordTest():void
    ■ sleep(long):void

```

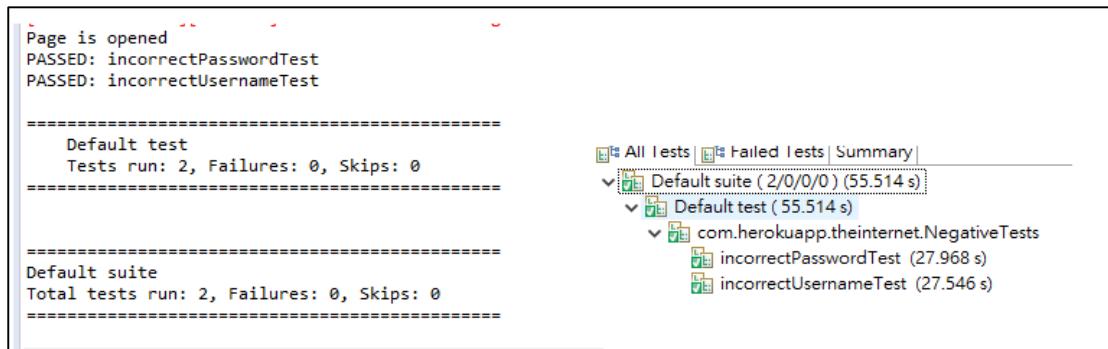
7. Run it

```

[INFO] Tests run: 1, Failures: 0, Skips: 0
=====
=====
Default suite
Total tests run: 1, Failures: 0, Skips: 0
=====
```

## 8. Remove comment @test for negative test, see what it will happen

It will run two test



```
Page is opened
PASSED: incorrectPasswordTest
PASSED: incorrectUsernameTest

=====
Default test
Tests run: 2, Failures: 0, Skips: 0
=====

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

All Tests | Failed Tests | Summary |  
Default suite (2/0/0/0) (55.514 s)  
  Default test (55.514 s)  
    com.herokuapp.theinternet.NegativeTests  
      incorrectPasswordTest (27.968 s)  
      incorrectUsernameTest (27.546 s)

## 9. Run Test under cmd

```
Page is opened
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 56.737 s - in com.herokuapp.theinternet.NegativeTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time:  59.941 s
[INFO] Finished at: 2019-12-21T22:19:35+08:00
[INFO]
```

# Part 3: Creating Test Suite-Login

## Account

## SetUP TestSuite-TestNG

### 1. Why do we need Test Suite?

The reason is because if you have many test case, we have to run individual, if we used testsuite, we can add many case in it. Basely Testsuite uses XML, it will run TestCase.

TestCase use java file

Test Suite use xml

## 2. How to combined test from different class?

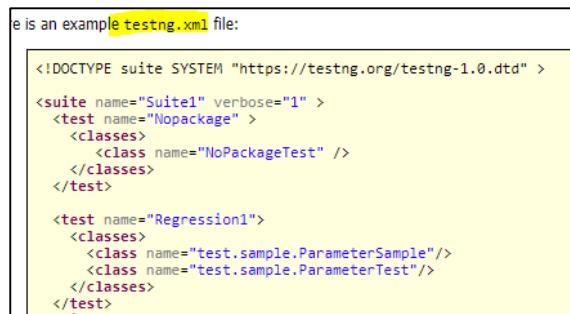
Go to this link for testing NG

<https://testng.org/doc/documentation-main.html>

## 3. Copy the “3 - testng.xml”

-Go to above page and click “3 - testng.xml”

-Copy the xml

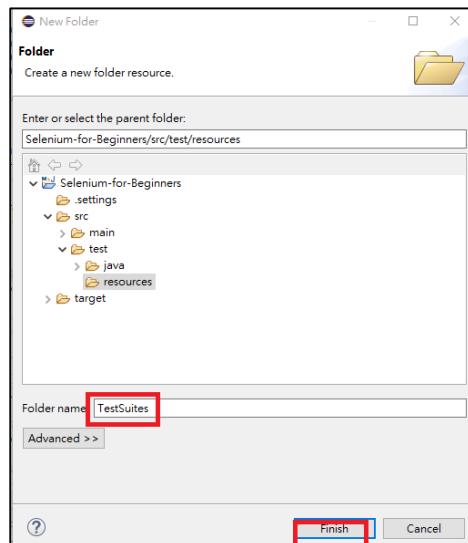
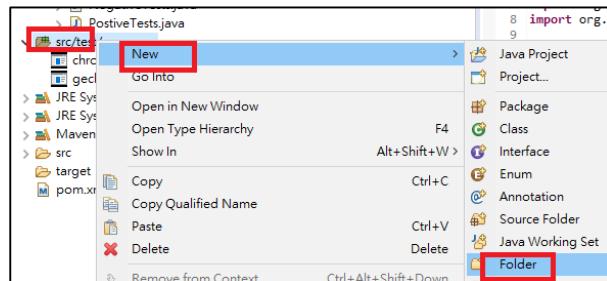


This screenshot shows a code editor displaying an example `testng.xml` file. The file content is as follows:

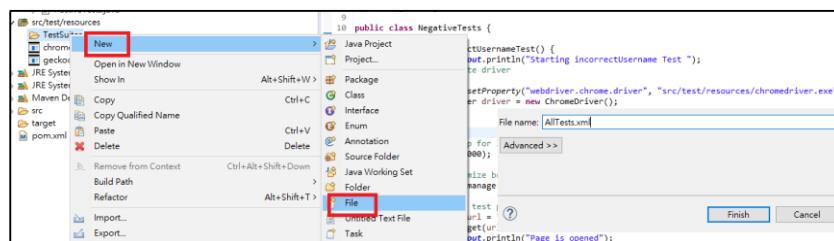
```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name="Suite1" verbose="1" >
  <test name="Nopackage" >
    <classes>
      <class name="NoPackageTest" />
    </classes>
  </test>

  <test name="Regression1" >
    <classes>
      <class name="test.sample.ParameterSample"/>
      <class name="test.sample.ParameterTest"/>
    </classes>
  </test>
```

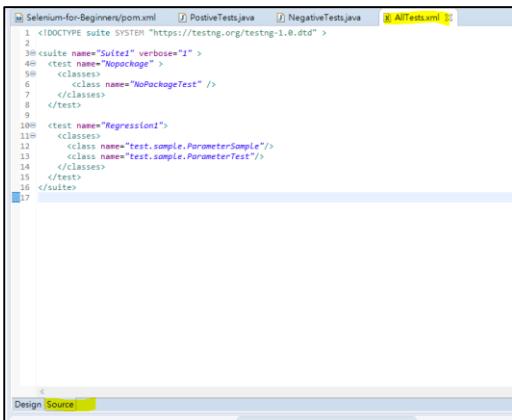
## 4. Add new folder under src/test/resources



## Add new File name AllTest.xml



Go to source, and paste the url testin.xml content into source



```
<!DOCTYPE suite SYSTEM "https://testing.org/testng-1.0.dtd">
<suite name="Suite1" verbose="1">
<test name="Mopckage">
<classes>
<class name="NoPackageTest" />
</classes>
</test>
<test name="Regression1">
<classes>
<class name="test.sample.ParameterSample"/>
<class name="test.sample.ParameterTest"/>
</classes>
</test>
</suite>
```

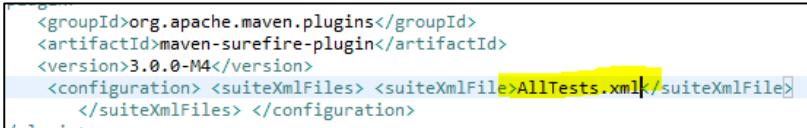
## 5. Edit Pom.xml file

Uncomment configuration



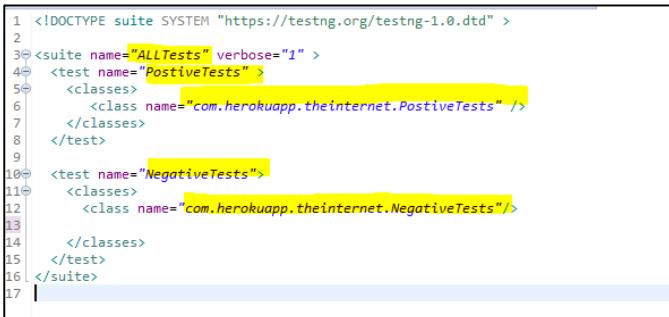
```
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-surefire-plugin</artifactId>
<version>3.0.0-M4</version>
<configuration> <suiteXmlFiles> <suiteXmlFile>testng.xml</suiteXmlFile>
</suiteXmlFiles> </configuration>
</plugin>
</plugins>
```

Changed the testing.xml to AllTests.xml



```
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-surefire-plugin</artifactId>
<version>3.0.0-M4</version>
<configuration> <suiteXmlFiles> <suiteXmlFile>AllTests.xml</suiteXmlFile>
</suiteXmlFiles> </configuration>
```

## 6. Modify AllTest.xml



```
<!DOCTYPE suite SYSTEM "https://testing.org/testng-1.0.dtd" >
<suite name="ALLTests" verbose="1" >
<test name="PositiveTests" >
<classes>
<class name="com.herokuapp.theinternet.PositiveTests" />
</classes>
</test>
<test name="NegativeTests" >
<classes>
<class name="com.herokuapp.theinternet.NegativeTests" />
</classes>
</test>
</suite>
```

Since we have 2 class, so we add two class positiveTests, and NegativeTests. Class name, put the full package name (you can find under properties).

# Test Suite1 Run Login Test

## Test Suite1-Run Login Success and Incorrect Account

Description: This Test is to Login wrong account name

File used:

PositiveTest.java: login success

NegativeTest.java: incorrect username and incorrect password

AllTest.xml: Test Suite

Folder name: case2-2-TestSuite

### 1. Running Test Suite

Changed only incorrectUsernameTest into Firefox driver, other just used chrome

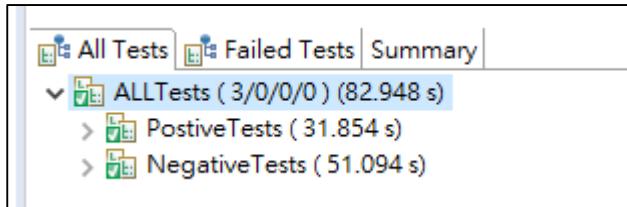
```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.Assert;
import org.testng.annotations.Test;

public class NegativeTests {
    @Test
    public void incorrectUsernameTest() {
        System.out.println("Starting incorrectUsername Test ");
        // create driver
        //System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
        System.setProperty("webdriver.gecko.driver", "src/main/resources/geckodriver.exe");
        //WebDriver driver = new ChromeDriver();
        WebDriver driver = new FirefoxDriver();
        // sleep for 3 second
        sleep(3000);
        // maximize browser window
        driver.manage().window().maximize();
    }
}
```

Run it right click Alltest.xml and run TestNG under Debug



```
=====
ALLTests
Total tests run: 3, Failures: 0, Skips: 0
=====
```



So you can see chrome and firefox is running below log

```

Page is opened
Starting incorrectPassword Test
Starting ChromeDriver 79.0.3949.16 (93fcc21110c10dbbd49bfff8f472335360e31d05-refs/branch-heads/3945@{#262}) on port 40702
Only local connections are allowed.
Please protect ports used by ChromeDriver and related test frameworks to prevent access by malicious code.
[1576971585.872] [WARNING]: Timed out connecting to Chrome, retrying...
12月 22, 2019 7:39:47 上午 org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
[1576971592.887] [WARNING]: Timed out connecting to Chrome, retrying...
Page is opened
Starting incorrectUsername Test
1576971611163 [mochirunner:runner] INFO Running command: "C:\Program Files\Mozilla Firefox\firefox.exe" "-marionette" "-foreground" "-no-remote" "-profile"
1576971611775 addons.webextension.screenshots@mozilla.org WARN Loading extension 'screenshots@mozilla.org': Reading manifest: Invalid extension permission
1576971611775 addons.webextension.screenshots@mozilla.org WARN Loading extension 'screenshots@mozilla.org': Reading manifest: Invalid extension permission
1576971611776 addons.webextension.screenshots@mozilla.org WARN Loading extension 'screenshots@mozilla.org': Reading manifest: Invalid extension permission
1576971611776 addons.webextension.screenshots@mozilla.org WARN Loading extension 'screenshots@mozilla.org': Reading manifest: Invalid extension permission
JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile directory.
[WARM]: compileToBinary(750).

```

## 2. Run under cmd method

We can used use this command “mvn clean test” to run all test

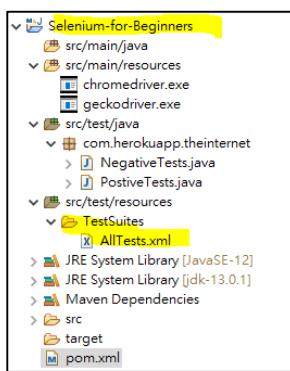
It will fail due to AllTest.xml file need to modifyit, marven don't know xml file

```

[INFO] -----
[ERROR] Suite file C:\Users\ChenChih\eclipse-workspace\Selenium-for-Beginners\AllTests.xml is not a valid file
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD FAILURE
[INFO]
[INFO] Total time: 3.261 s
[INFO] Finished at: 2019-12-22T07:45:34+08:00
[INFO]
[INFO] [ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:3.0.0-M4:test (default-test) on project Selenium-for-Beginners: There are test failures.
[INFO] [ERROR] Please refer to C:\Users\ChenChih\eclipse-workspace\Selenium-for-Beginners\target\surefire-reports for the individual test results.
[INFO] [ERROR] Please refer to dump files (if any exist) [date].dump, [date]-jvmRun[N].dump and [date].dumpstream.

```

Under the Selenium-for-Beginners this path that we execute mvn-clean test it could not find AllTest.xml.



The reason is pom.xml we just put AllTest.xml, but it could not find it, it because AllTests.xml is located in different location.

Just need to give full path of AllTests.xml to it, as below:

Change AllTest.xml to `src/test/resources/TestSuites/AllTests.xml`

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>3.0.0-M4</version>
  <configuration> <suiteXmlFiles> <suiteXmlFile>src/test/resources/TestSuites/AllTests.xml</suiteXmlFile>
    </suiteXmlFiles> </configuration>
  </plugin>
</plugins>
```

```
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 81.76 s - in TestSuite
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:24 min
[INFO] Finished at: 2019-12-22T08:03:49+08:00
[INFO] -----
```

C:\Users\ChenChih\eclipse-workspace\Selenium-for-Beginners>

## TestingNG-Annotations:@Test-Priority

This Test is to learn how to used `@Test(priority)`, if you type in 1, this will run first, if no type default will run first.

Test File:

### Login Incorrect account and password test

Description: This Test is to Login wrong account using priority parameter

#### File used:

PositiveTest.java: login success

NegativeTest.java: incorrect username and incorrect password

AllTest.xml: Test Suite

Folder name: case3-1-TestNG-Priority

1. Go to TestNG link

Test NG link: <https://testng.org/doc/documentation-main.html>

2. Used `priority` this parameter for `@test()`

We can add some variable under `@test()` for which priority

3. go to the link and Press “2 – Annotations”

<b>@Test</b>	Marks a class or a method as part of the test.
alwaysRun	If set to true, this test method will always be run even if it depends on a method that failed.
dataProvider	The name of the data provider for this test method.
dataProviderClass	The class where to look for the data provider. If not specified, the data provider will be looked on the class of the current test method or one of its base classes. If this attribute is specified, the class where the provider is located will be used.
dependsOnGroups	The list of groups this method depends on.
dependsOnMethods	The list of methods this method depends on.
description	The description for this method.
enabled	Whether methods on this class/method are enabled.
expectedExceptions	The list of exceptions that a test method is expected to throw. If no exception or a different than one on this list is thrown, this test will be marked a failure.
groups	The list of groups this class/method belongs to.
invocationCount	The number of times this method should be invoked.
invocationTimeout	The maximum number of milliseconds this test should take for the cumulated time of all the invocationcounts. This attribute will be ignored if invocationCount is not specified.
<b>priority</b>	The priority for this test method. Lower priorities will be scheduled first.
successPercentage	The percentage of success expected from this method
singleThreaded	If set to true, all the methods on this test class are guaranteed to run in the same thread, even if the tests are currently being run with parallel="methods". This attribute can only be used at the class level.
timeOut	The maximum number of milliseconds this test should take.
threadPoolSize	The size of the thread pool for this method. The method will be invoked from multiple threads as specified by invocationCount. Note: this attribute is ignored if invocationCount is not specified

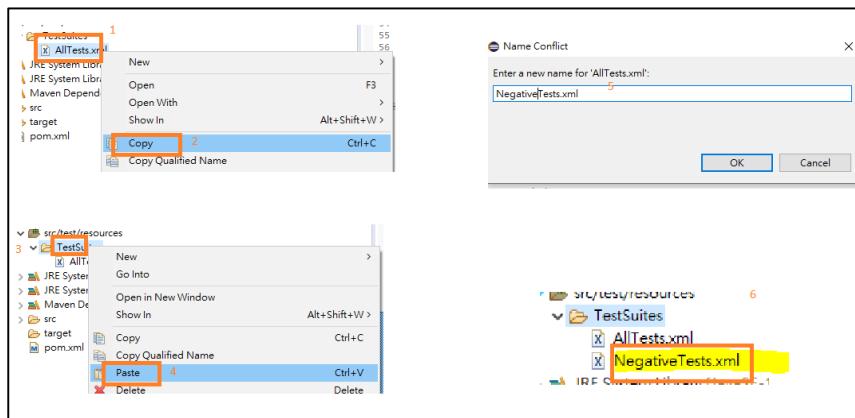
#### 4. Edit negativeTest.java

```
public class NegativeTests {
    @Test(priority =1)
    public void incorrectUsernameTest() {
        System.out.println("Starting incorrectUsername Test ");
        // create driver
    }
}
```

```
    @Test(priority =2)
    public void incorrectPasswordTest() {
        System.out.println("Starting incorrecPassword Test ");
        // create driver
        System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");

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

#### 5. Copy AllTest.xml file again and rename negativetest.xml



## 6. Modify NegativeTests.xml

```
1 <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
2
3<@suite name="NegativeTestsSuite" verbose="1">
4
5@<test name="NegativeTests">
6@<classes>
7    <class name="com.herokuapp.theinternet.NegativeTests" />
8
9</classes>
10</test>
11</suite>
12
```

## 7. Run NegativeTest.xml

```
Starting incorrectPassword Test
Starting ChromeDriver 79.0.3945.16 (93fcc21110c10dbbd49bbff8f472335360e31d05-refs/branch-heads/3945@)
Only local connections are allowed.
Please protect ports used by ChromeDriver and related test frameworks to prevent access by malicious
[1576975167.844][WARNING]: Timed out connecting to Chrome, retrying...
12月 22, 2019 8:39:29 上午 org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
[1576975174.864][WARNING]: Timed out connecting to Chrome, retrying...
Page is opened
=====
NegativeTestsSuite
Total tests run: 2, Failures: 0, Skips: 0
=====
```

# TestingNG-Annotations:@Test-Enable

## Login Incorrect username Test

The purpose of this is when release a code, some code will be broken but we still want to run regression test, we can enable or disable it

Description: This Test is to Login wrong account using enable parameter

### File used:

PositiveTest.java: login success

NegativeTests\_2\_enable.java: incorrect username and incorrect password

AllTest.xml: Test Suite

Folder name: case3-1-TestNG-Priority

enabled	Whether methods on this class/method are enabled.
expectedExceptions	The list of exceptions that a test method is expected to throw. If no exceptions are expected, leave this empty.

### 1. Modify negativeTest.java

set @Test to disable like this enabled=false

```
@Test(priority =2, enabled =false)
public void incorrectPasswordTest() {
    System.out.println("Starting incorrectPassword Test ");
    // create driver
    WebDriver driver = new FirefoxDriver();
    driver.get("https://the-internet.herokuapp.com/login");
    driver.manage().window().maximize();
```

## 2. Run the test

So only one test will be run

```
PASSED: incorrectUsernameTest
=====
Default test
Tests run: 1, Failures: 0, Skips: 0
=====

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

# TestingNG-Annotations:@Test Groups

## Login Incorrect username Test

Description: This Test is to Login wrong account using groups

### File used:

NegativeTests.java: incorrect username and incorrect password

NegativeTests-group.xml: Test Suite

Folder name: case3-2-TestNG-Group-include

### 1. Modify negativeTest.java

Add smokeTests to incorrectUsernameTest, but for incorrectpassword, don't put group test yet.

smokeTests define by our-self

```
public class NegativeTests {
    @Test(priority =1, groups= { "negativeTests", "smokeTests" })
    public void incorrectUsernameTest() {
        System.out.println("Starting incorrectUsername Test ");
        // create driver
    }

    @Test(priority =2, enabled =false, groups= { "negativeTests" })
    public void incorrectPasswordTest() {
        System.out.println("Starting incorrecPassword Test ");
        // create driver
    }
}
```

### 2. Modify NegativeTests.xml

Go to documentation and copy xml file

```
Invoking TestNG with
<test name="Test1">
    <groups>
        <run>
            <include name="functest"/>
        </run>
    </groups>
    <classes>
        <class name="example1.Test1"/>
    </classes>
</test>
```

Copy to NegativeTests.xml, and changed group to “smokeTests”

```
2
3<suite name="NegativeTestsSuite" verbose="1">
4
5    <test name="NegativeTests">
6        <groups>
7            <run>
8                <include name="smokeTests" />
9            </run>
10           </groups>
11
12       <classes>
13           <class name="com.herokuapp.theinternet.NegativeTests" />
14
15       </classes>
16   </test>
17 </suite>
18
19
```

### 3. Enable incorrectPassword to enable

Remove `enabled =false`

```
@Test(priority =2, groups= { "negativeTests" })
public void incorrectPasswordTest() {
    System.out.println("Starting incorrectPassword Test ");
    // create driver
    System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver");
}
```

So both function will look like this

```
public class NegativeTests {
    @Test(priority =1, groups= { "negativeTests", "smokeTests" })
    public void incorrectUsernameTest() {
        System.out.println("Starting incorrectUsername Test ");

    }

    @Test(priority =2, groups= { "negativeTests" })
    public void incorrectPasswordTest() {
        System.out.println("Starting incorrectPassword Test ");
    }
}
```

### 4. Run NegativeTests.xml and will only run incorrectUsernameTest

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

All Tests Failed Tests | Summary |
v [ ] NegativeTestsSuite (1/0/0) (22.906 s)
  v [ ] NegativeTests (22.906 s)
    v [ ] com.herokuapp.theinternet.NegativeTests
      [ ] incorrectUsernameTest (22.906 s)
```

# TestingNG- Test Method groups

## (include/exclude)

### Login Incorrect username and password

Description: This Test is to Login wrong account using include parameter

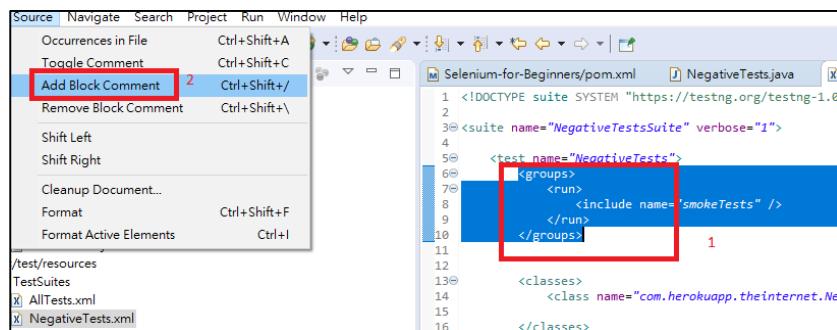
#### File used:

NegativeTests.java: incorrect username and incorrect password

NegativeTests-include.xml: Test Suite

Folder name: case3-2-TestNG-Group-include

#### 1. Block the group in NegativeTests.xml



#### 3. Change to include or exclude

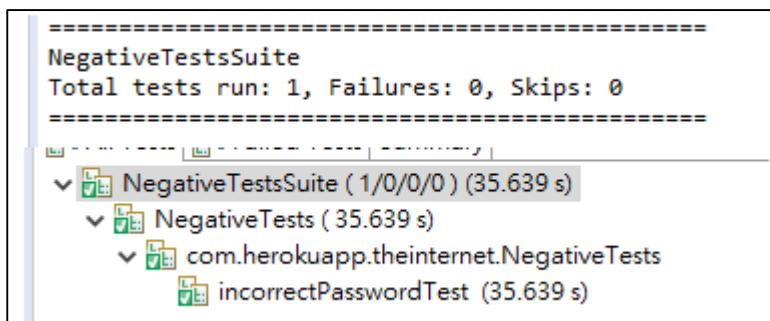
Add name to “incorrectPasswordTest”

```
<test name="NegativeTests">
  <!-- <groups> <run> <include name="smokeTests" /> </run> </groups> -->

  <classes>
    <class name="com.herokuapp.theinternet.NegativeTests">
      <methods>
        <include name="incorrectPasswordTest" />
      </methods>
    </class>
  </classes>
</test>
</suite>
```

#### 4. Run the test

IncorrectPasswordTest will be executed

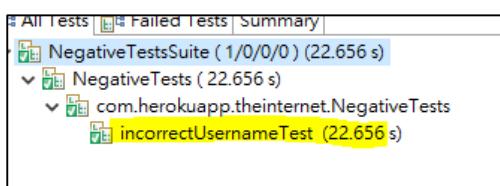


#### 5. Change to exclude

```
<exclude name="incorrectPasswordTest" />
```

#### 6. Run the test

7. After Run incorrectPasswordTest will not run



## TestingNG: @Parameters

### Login in Correct account and password

Description: This Test is to Login wrong account using passing parameter

#### File used:

NegativeTests.java:

NegativeTests.xml: Test Suite

Folder name: case4-1-passparamater

## Pass parameter to test method

## 1. Modify NegativeTest.xml

Remove Group, and method tag, and changed Test name to NegativeUsernameTests.

```
1 <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
2
3<suite name="NegativeTestsSuite" verbose="1">
4
5<test name="NegativeUsernameTests">
6    <parameter name="username" value="IncorrectUsername" />
7    <parameter name="password" value="SuperSecretPassword!" />
8    <parameter name="expectedMessage" value="Your username is invalid!" />
9    <classes>
10        <class name="com.herokuapp.theinternet.NegativeTests">
11            </class>
12        </classes>
13    </test>
14
15<test name="NegativePasswordTests">
16    <parameter name="username" value="tomsmith" />
17    <parameter name="password" value="IncorrectPassword!" />
18    <parameter name="expectedMessage" value="Your password is invalid!" />
19    <classes>
20        <class name="com.herokuapp.theinternet.NegativeTests">
21            </class>
22        </classes>
23    </test>
24</suite>
```

## 2. Modify negativetest.java

comment incorrectPasswordTest this function, left with only one function

```

  //test(priority <= 1) groups { "negativeTests" } public void
  //negativeTest() {
  System.out.println("Starting incorrectPassword Test"); // create driver
  System.setProperty("webdriver.chrome.driver",
  "src/main/resources/chromedriver.exe");

  WebDriver driver = new ChromeDriver();
  // maximize browser window driver.manage().window().maximize();

  // open test page String url = "https://the-internet.herokuapp.com/login";
  driver.get(url); System.out.println("Page is opened");
  // sleep for 2 second sleep(2000);

  // enter username WebElement username =
  driver.findElement(By.id("username")); username.sendKeys("tomsmith");
  sleep(1000); // enter password WebElement password =
  driver.findElement(By.id("password"));
  password.sendKeys("IncorrectPassword!"); sleep(2000); // click login button
  WebElement loginbutton = driver.findElement(By.tagName("button"));
  loginbutton.click(); sleep(2000); // check error message
  String errorMessage = driver.findElement(By.id("flash")); String expectedErrorMessage =
  "Your password is invalid!";
  assertEquals(errorMessage, expectedErrorMessage);
  Assert.assertEquals(errorMessage, expectedErrorMessage);
  // Assert.assertEquals(errorMessage, expectedErrorMessage);
  // Actual Error Message did not contain expected. In fact it is:
  // "Your password is invalid!" + expectedErrorMessage); // close browser
  driver.quit(); }

```

After we save, we will only see incorrectUsernameTest this function

The screenshot shows the PyCharm IDE's Outline view. The tree structure displays the following elements:

- com.herokuapp.theinternet
- NegativeTests
  - incorrectUsernameTest(): void
  - sleep(long): void

### 3. Changed Method incorrectUsernameTest to NegativeLoginTest

```
public class NegativeTests {  
    @Test(priority =1, groups= { "negativeTests", "smokeTests" })  
    public void NegativeLoginTest() {  
        System.out.println("Starting incorrectUsername Test ");  
        // create driver
```

#### 4. Add @Parameter

Add Parameter "@Parameters({ "first-name" })", so since xml have three parameter, username, password, and expectedMessage, we need to add three parameter. So it should look like this:

```
@Parameters({ "username", "password", "expectedMessage" })
```

Also import as below:

```
import org.testng.annotations.Parameters;
```

```
public class NegativeTests {
    @Parameters({ "username", "password", "expectedMessage" })

    @Test(priority = 1, groups = { "negativeTests", "smokeTests" })
    public void NegativeLoginTest() {
        System.out.println("Starting incorrectUsername Test ");
        // create driver
    }
}
```

Also add three parameter to the NegativeLoginTest(), like this:

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

public class NegativeTests {
    @Parameters({ "username", "password", "expectedMessage" })

    @Test(priority = 1, groups = { "negativeTests", "smokeTests" })
    public void NegativeLoginTest(String username, String password, String expectedErrorMessage) {
        System.out.println("Starting incorrectUsername Test ");
        // create driver
        //System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
    }
}
```

#### 5. Changed WebElement

We have to change from WebElement username to usernameElement, is because the type there is duplicate name, so we have to change different name.

```
@Parameters({ "username", "password", "expectedMessage" })
@Test(priority = 1, groups = { "negativeTests", "smokeTests" })
public void negativeLoginTest(String username, String password, String expectedErrorMessage) {
    System.out.println("Starting incorrectUsernameTest");

    // Create driver
    System.setProperty("webdriver.gecko.driver", "src/main/resources/geckodriver");
    WebDriver driver = new FirefoxDriver();

    // sleep for 3 seconds
    sleep(3000);

    // maximize browser window
    driver.manage().window().maximize();

    // open test page
    String url = "http://the-internet.herokuapp.com/login";
    driver.get(url);
    System.out.println("Page is opened.");

    // action on LoginPage
    WebElement usernameElement = driver.findElement(By.id("username"));
    usernameElement.sendKeys("incorrectUsername");
```

## 6. Full Code

```
1  import org.openqa.selenium.*;
2
3  public class NegativeTests {
4      @Parameters({ "username", "password", "expectedMessage" })
5
6      @Test(priority = 1, groups = { "negativeTests", "smokeTests" })
7      public void NegativeLoginTest(String username, String password, String expectedErrorMessage) {
8          System.out.println("Starting negativeLogin Testwith "+username+"and"+password);
9          // create driver
10
11         //System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
12         //System.setProperty("webdriver.gecko.driver", "src/main/resources/geckodriver.exe");
13         WebDriver driver = new ChromeDriver();
14         //WebDriver driver = new FirefoxDriver();
15         // sleep for 3 second
16         sleep(3000);
17
18         // maximize browser window
19         driver.manage().window().maximize();
20
21         // open test page
22         String url = "http://the-internet.herokuapp.com/login";
23         driver.get(url);
24         System.out.println("Page is opened");
25
26         // sleep for 2 second
27         sleep(2000);
28
29         // enter username
30         WebElement usernameElement = driver.findElement(By.id("username"));
31         //usernameElement.sendKeys("incorrectUsername");
32         usernameElement.sendKeys(username);
33         sleep(1000);
34
35         // enter password
36         WebElement passwordElement = driver.findElement(By.name("password"));
37         //passwordElement.sendKeys("SuperSecretPassword!");
38         passwordElement.sendKeys(password);
39         sleep(3000);
40
41         // click login button
42         WebElement loginButton = driver.findElement(By.tagName("button"));
43         loginButton.click();
44         sleep(5000);
45
46         //verification
47         WebElement ErrorMessage = driver.findElement(By.id("flash"));
48         //String expectedErrorMessage = "Your username is invalid!";
49         String actualErrorMessage=ErrorMessage.getText();
50         Assert.assertTrue(actualErrorMessage.contains(expectedErrorMessage),
51             "Actual Error Message does not contain expected.\nActual: " + actualErrorMessage
52     }
53 }
```

## 7. Run TestSuite

```
=====
NegativeTestsSuite
Total tests run: 2, Failures: 0, Skips: 0
=====
  ↳ All Tests | Failed Tests | Summary |
    ↳ NegativeTestsSuite (2/0/0/0) (45.362 s)
      > ↳ NegativeUsernameTests (22.873 s)
      > ↳ NegativePasswordTests (22.489 s)
```

# Test Suite2 TestingNG- combine LoginTest

## Test Suite 2- Run Login Success and Incorrect Account

Description: This Test is to combine all variable what we learn before

File used:

LoginTests.java

LoginTests.xml

Folder name: Case5-1\_AllTest\_login

1. Copy NegativeTests.java(NegativeLoginTest function) to PostiveTest.java

So Negative class will become like this:

```

1 package com.herokuapp.theinternet;
2
3 import org.openqa.selenium.By;
4 import org.openqa.selenium.WebDriver;
5 import org.openqa.selenium.WebElement;
6 import org.openqa.selenium.chrome.ChromeDriver;
7 import org.openqa.selenium.firefox.FirefoxDriver;
8 import org.junit.Assert;
9 import org.junit.Parameters;
10 import org.junit.Test;
11
12 public class NegativeTests {
13
14
15     /*
16      * @Test(priority = 2, groups= { "negativeTests" }) public void
17      * incorrectpasswordTest()
18      * {
19      * System.out.println("Starting incorrectPassword Test "); // create driver
20      * System.setProperty("webdriver.chrome.driver",
21      * "src/main/resources/chromedriver.exe");
22      * WebDriver driver = new ChromeDriver();
23      *
24      * // sleep for 3 second sleep(3000);
25      *
26      * // maximize browser window driver.manage().window().maximize();
27      *
28      * // open test page String url = "http://the-internet.herokuapp.com/login";
29      * driver.get(url); System.out.println("Page is opened");
30      *
31      * // sleep for 2 second sleep(2000);
32      */

```

## 2. Paste to PostiveTest.java

```

@Parameters({ "username", "password", "expectedMessage" })
@Test(priority = 1, groups= { "negativeTests", "smokeTests" })
public void NegativeLoginTest(String username, String password, String expectedErrorMessage) {
    System.out.println("Starting negativeLogin Test with "+username+" and "+password);
    // create driver

    //System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
    System.setProperty("webdriver.gecko.driver", "src/main/resources/geckodriver.exe");
    //WebDriver driver = new ChromeDriver();
    WebDriver driver = new FirefoxDriver();
    // sleep for 3 second
    sleep(3000);

    // maximize browser window
    driver.manage().window().maximize();

    // open test page
    String url = "http://the-internet.herokuapp.com/login";
    driver.get(url);
    System.out.println("Page is opened");

    // sleep for 2 second
    sleep(2000);

    // enter username
    WebElement usernameElement = driver.findElement(By.id("username"));
    //usernameElement.sendKeys("incorrectUsername");
    usernameElement.sendKeys(username);
    sleep(1000);
}

```

## 3. Modify PostiveTest.java

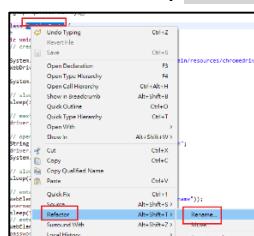
Rename `public void loginTest()` to `public void PostiveLoginTest()`

```

12 public class PositiveTests {
13     @Test
14     public void PostiveLoginTest() {
15         // create driver

```

## 4. Rename file positiveTests to LoginTest



You can used refactor, it will automatic help you change file name and function name also

The screenshot shows a Java code editor with the following code:

```

public class XXXX {
    @Test
    public void PositiveLoginTest() {
        // create driver
    }
}
```

A tooltip at the bottom says "Press Enter to refactor. Options...". To the right, a list of refactoring options is shown:

- >  **LoginTests.java**
- >  **NegativeTests\_backup.java**
- >  **NegativeTests.java**
- >  **PositiveTests\_backup.java**

The "LoginTests.java" option is highlighted with a yellow box.

5. We can delete NegativeTest.java if you want

6. Code LoginTest.java as below:

```
11
12 public class LoginTests {
13     @Test(priority =1, groups= { "positiveTests", "smokeTests" })
14     public void PositiveLoginTest() {
15         // create driver
16
17         System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
18         WebDriver driver = new ChromeDriver();
19
20         System.out.println("Starting login page ");
21 }
```

Go to negative.java and copy function: NegativeLoginTest to LoginTest.java as below:

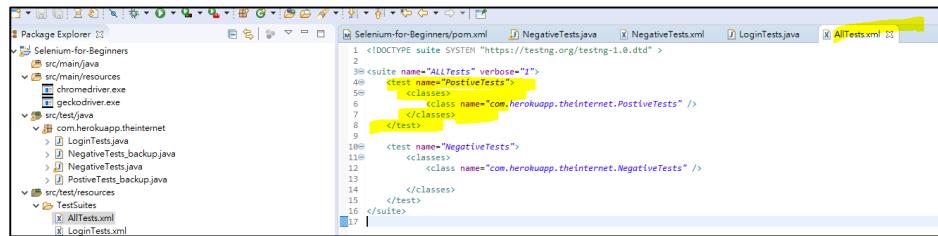
```
@Parameters({ "username", "password", "expectedMessage" })
@Test(priority =2, groups= { "negativeTests", "smokeTests" })
public void NegativeLoginTest(String username, String password, String expectedErrorMessage) {
    System.out.println("Starting negativeLogin Testwith "+username+"and"+password);
    // create driver
}
```

We have to import `import org.testng.annotations.Parameters;`, else will have problem for `@Parameters({ "username", "password", "expectedMessage" })`

7. Create new xml suite LoginTest.xml (copy NegativeTests.xml)

Copy AllTests.xml's positiveTest.

AllTests.xml



LoginTest.xml

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="NegativeTestsSuite" verbose="1">
<test name="PositiveTests">
<classes>
<class name="com.herokuapp.theinternet.PostiveTests" />
</classes>
</test>
<test name="NegativeUsernameTests">
<parameters>
<parameter name="username" value="incorrectUsername" />
<parameter name="password" value="SuperSecretPassword!" />
<parameter name="expectedMessage"
           value="Your username is invalid!" />
</parameters>
<classes>
<class name="com.herokuapp.theinternet.NegativeTests" />
</classes>
</test>
<test name="NegativePasswordTests">
<parameters>
<parameter name="username" value="tomsmith" />
<parameter name="password" value="incorrectPassword!" />
<parameter name="expectedMessage"
           value="Your password is invalid!" />
</parameters>
<classes>
<class name="com.herokuapp.theinternet.NegativeTests" />
</classes>
</test>

```

We can also delete AllTest.xml if you like

## 8. Add group to positiveTests

```
<suite name="NegativeTestsSuite" verbose="1">
  <test name="PositiveTests">
    <groups>
      <run>
        <include name="positiveTests"></include>
      </run>
    </groups>
    <classes>
      <class name="com.herokuapp.theinternet.LoginTests" />
    </classes>
  </test>
```

## 9. Add method to NegativeUsernameTests, and NegativePasswordTests

These two method are the same

```
<test name="NegativeUsernameTest">
  <parameter name="username" value="incorrectUsername" />
  <parameter name="password" value="SuperSecretPassword!" />
  <parameter name="expectedMessage"
    value="Your username is invalid!" />

  <classes>
    <class name="com.herokuapp.theinternet.LoginTests">
      <methods>
        <include name="negativeLoginTest">
      </include>
    </methods>
  </class>
```

Please changed to LoginTests like this: <com.herokuapp.theinternet.LoginTests>

Else will fail

## 10. Run it

```
=====
NegativeTestsSuite
Total tests run: 3, Failures: 0, Skips: 0
=====
```

## 11. Summary code

```
-----  
LoginTest.xml  
-----  
1 <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >  
2 <suite name="NegativeTestsSuite" verbose="1">  
3   <test name="PositiveTests">  
4     <groups>  
5       <run>  
6         <include name="positiveTests"></include>  
7       </run>  
8     </groups>  
9     <classes>  
10       <class name="com.herokuapp.theinternet.LoginTests" />  
11     </classes>  
12   </test>  
13  
14  
15   <test name="NegativeUsernameTest">  
16     <parameter name="username" value="incorrectUsername" />  
17     <parameter name="password" value="SuperSecretPassword!" />  
18     <parameter name="expectedMessage"
19       value="Your username is invalid!" />  
20  
21     <classes>  
22       <class name="com.herokuapp.theinternet.LoginTests">
23         <methods>
24           <include name="negativeLoginTest">
25         </include>
26       </methods>
27     </class>
28   </classes>
29 </test>  
30  
31 <start name="NegativePasswordTest">
```

```
-----  
LoginTest.java  
-----  
1 package com.herokuapp.theinternet;  
2  
3 import org.openqa.selenium.By;  
4  
5 public class LoginTests {  
6   @Test(priority = 1, groups = { "positiveTests", "smokeTests" })  
7   public void PostiveLogonTest() {  
8     // create driver  
9  
10    System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver");  
11    WebDriver driver = new ChromeDriver();  
12  
13    System.out.println("Starting positiveLogon Test with " + username + " and " + password);  
14    // create driver  
15  
16    System.setProperty("webdriver.gecko.driver", "src/main/resources/geckodriver");  
17    WebDriver driver = new FirefoxDriver();  
18  
19    // sleep for 3 second  
20    sleep(3000);  
21  
22    // maximize browser window  
23    driver.manage().window().maximize();  
24  
25    // open test page  
26    String url = "http://the-internet.herokuapp.com/login";  
27  
28    driver.get(url);  
29  
30    // verify title  
31    assertEquals("The Internet", driver.getTitle());  
32  
33    // close browser  
34    driver.quit();  
35  }
```

# Part 3-2: Creating Test Suite-Driven

# and Browser

# Test Suite3 TestingNG- Annotations-

# BeforeMethod and AfterMethod

# Test Suite 3 Add Setup method for Driver

<https://testng.org/doc/documentation-main.html#annotations>

<b>@BeforeSuite</b>	The annotated method will be run before all tests in this suite have run.
<b>@AfterSuite</b>	The annotated method will be run after all tests in this suite have run.
<b>@BeforeTest</b>	The annotated method will be run before any test method belonging to the classes inside the <test> tag is run.
<b>@AfterTest</b>	The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run.
<b>@BeforeGroups</b>	The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method.
<b>@AfterGroups</b>	The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method.
<b>@BeforeClass</b>	The annotated method will be run before the first test method in the current class is invoked.
<b>@AfterClass</b>	The annotated method will be run after all the test methods in the current class have been run.
<b>@BeforeMethod</b>	The annotated method will be run before each test method.
<b>@AfterMethod</b>	The annotated method will be run after each test method.
<b>Behaviour of annotations in superclass of a TestNG class</b>	

Description: This Test is to add method for run put driver first

## **File used:**

## LoginTests.java: incorrect username and incorrect password

## LoginTests.xml: Test Suite

Folder name: Case5-2\_before\_after\_method

1. Create a new private `setUp` function `private void setUp()`

And copy driver code to this function

```
before

greet(priority = 1, groups = [ "positiveTests", "smokeTests" ])
public void PositiveLoginTest() {
    // create driver
    System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    // sleep for 3 second
    sleep(3000);
    // maximize browser window
    driver.manage().window().maximize();
    // open test page
    String url = "http://the-internet.herokuapp.com/login";
    driver.get(url);
    System.out.println("Page is opened");

    after

    public class loginTests {
        @Test
        private void setup() {
            // create driver
            System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
            WebDriver driver = new ChromeDriver();
            driver.manage().window().maximize();
            // sleep for 3 second
            sleep(3000);
            // maximize browser window
            driver.manage().window().maximize();
        }
        @Test(priority = 1, groups = [ "positiveTests", "smokeTests" ])
        public void PositiveLoginTest() {
            // open test page
            String url = "http://the-internet.herokuapp.com/login";
            driver.get(url);
            System.out.println("Page is opened");
        }
    }
}
```

## 2. Delete the driver information in negativetest

Since we have already add setup function, so we can remove driver information in negativetest related function.

```
@Parameters({ "username", "password", "expectedMessage" })
@Test(priority = 2, groups = { "negativeTests", "smokeTests" })
public void negativeLoginTest(String username, String password, String expectedErrorMessage) {
    System.out.println("Starting negativeLogin Testwith " + username + "and" + password);
    // create driver
    // System.setProperty("webdriver.chrome.driver",
    // "src/main/resources/chromedriver.exe");
    // System.setProperty("webdriver.gecko.driver", "src/main/resources/geckodriver.exe");
    // WebDriver driver = new ChromeDriver();
    // sleep for 3 second
    sleep(3000);
    // maximize browser window
    driver.manage().window().maximize();

    // open test page
    String url = "http://the-internet.herokuapp.com/login";
    driver.get(url);
    System.out.println("Page is opened");

    // sleep for 2 second
    sleep(2000);
}

@Parameters({ "username", "password", "expectedMessage" })
@Test(priority = 2, groups = { "negativeTests", "smokeTests" })
public void negativeLoginTest(String username, String password, String expectedErrorMessage) {
    System.out.println("Starting negativeLogin Testwith " + username + "and" + password);
}

// open test page
String url = "http://the-internet.herokuapp.com/login";
driver.get(url);
System.out.println("Page is opened");

// sleep for 2 second
```

## 3. add @BeforeMethod as below:

The screenshot shows the Java code for LoginTests. At line 13, there is a red box around the annotation `@BeforeMethod`. A tooltip appears with the following options:

- Import 'BeforeMethod' (org.testng.annotations)
- Add TestNG library
- Create annotation 'BeforeMethod'
- Change to 'BeforeTest' (org.testng.annotations)
- Rename in file (Ctrl+2, R)
- Convert to TestNG (Annotations)
- Outline @Test annotations to the class level

On the right side of the code editor, there is a preview pane showing the imports for the annotation:

```
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.Assert;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Parameters;
```

## 4. Check Problem

The screenshot shows the Java code for LoginTests. There are several red error markers (exclamation points) scattered throughout the code, indicating unresolved variable issues. The code includes setup steps like opening a URL, sending keys to input fields, clicking a button, and performing a verification step. Below the code editor, the 'Problems' view shows the following errors:

Description	Resource	Path	Location	Type
driver cannot be resolved	LoginTests.java	/Selenium-for-Be...	line 37	Java Problem
driver cannot be resolved	LoginTests.java	/Selenium-for-Be...	line 44	Java Problem
driver cannot be resolved	LoginTests.java	/Selenium-for-Be...	line 48	Java Problem
driver cannot be resolved	LoginTests.java	/Selenium-for-Be...	line 52	Java Problem
driver cannot be resolved	LoginTests.java	/Selenium-for-Be...	line 58	Java Problem
driver cannot be resolved	LoginTests.java	/Selenium-for-Be...	line 63	Java Problem

Driver can't use cause there's no this variable so we need to create it by:

Move webdriver driver this variable and place under LoginTest and create a private variable as like this `private WebDriver driver;`

```

1 public class LoginTests {
2     @BeforeMethod
3     private void setup() {
4         // create driver
5         System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
6         WebDriver driver = new ChromeDriver();
7         System.out.println("Starting login page ");
8         // sleep for 3 second
9         sleep(3000);
10        // maximize browser window
11        driver.manage().window().maximize();
12    }
13
14    @Test
15    public void test() {
16        // create driver
17        System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
18        WebDriver driver = new ChromeDriver();
19        System.out.println("Starting login page ");
20        // sleep for 3 second
21        sleep(3000);
22        // maximize browser window
23    }
24
25    @AfterMethod
26    public void teardown() {
27        // close browser
28    }
29

```

5. After saving it, look at problem, just left with warning, this is related to Firefox, just remove import firefox, then it will be fine

Description	Resource	Path	Location	Type
The import org.openqa.selenium.WebDriver is never used	LoginTests.java	/Selenium-for-B.../line 3	Java Problem	
The import org.openqa.selenium.firefox.FirefoxDriver is never used	LoginTests.java	/Selenium-for-B.../line 4	Java Problem	
The import org.openqa.selenium.chrome.ChromeDriver is never used	LoginTests.java	/Selenium-for-B.../line 5	Java Problem	
The import org.openqa.selenium.WebDriver is never used	NegativeTests.java	/Selenium-for-B.../line 6	Java Problem	
The import org.openqa.selenium.firefox.FirefoxDriver is never used	NegativeTests.java	/Selenium-for-B.../line 7	Java Problem	
The import org.openqa.selenium.WebDriver is never used	NegativeTests.java	/Selenium-for-B.../line 8	Java Problem	
The import org.openqa.selenium.WebElement is never used	NegativeTests.java	/Selenium-for-B.../line 9	Java Problem	

6. Run with Fail

```

Starting negativeLogin Testwith tomsmithandincorrectPassword!
Page is opened
=====
NegativeTestsSuite|
Total tests run: 3, Failures: 1, Skips: 0
=====
```

7. Fail and explanation

Search:	Passed: 2	Failed: 1	Skipped: 0	Total: 3/3 Methods: 3 (56410 ms)
All Tests   Failed Tests   Summary				
↳ NegativeTestsSuite (2/1/0.0) (0.796 s)				
↳ PositiveTests (0.012 s)				
↳ com.herokuapp.theinternet.LoginTests				
↳ PostiveLoginTest (0.012 s)				
↳ NegativeUsernameTest (15.528 s)				
↳ NegativePasswordTest (15.256 s)				

Failure Exception  
java.lang.NullPointerException  
at com.herokuapp.theinternet.LoginTests.PostiveLoginTest(LoginTests.java:39)  
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl)  
at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl)  
at org.testng.remote.AbstractRemoteTestNG.run(AbstractRemoteTestNG.java:115)

Driver will run first, then run method

```

[RemoteTestNG] detected TestNG version 6.14.3
[TestingContentHandler] [WARN] It is strongly recommended to add "<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd ">" at the top of your file, otherwise TestNG Starting login page
Starting ChromeDriver 79.0.3945.16 (95e5dcf67cda9b8a53787c0a7235360e31d05-refs/branch-heads/3945@{#422}) on port 10536
Only local connections are allowed.
Please protect ports used by ChromeDriver and related test frameworks to prevent access by malicious code.
[1577574763.289][WARNING]: Timed out connecting to Chrome, retrying...
INFO: Detected dialect: WBC
[1577574770.335][WARNING]: Timed out connecting to Chrome, retrying...
Starting negativeLogin Testwith incorrectUsernameandSuperSecretPassword!
Page is opened
=====
test
driver
```

Driver will run first due to we add the Before Method

```
private WebDriver driver;
private String url;
private void setup() {
    // create driver
    System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
    driver = new ChromeDriver();
}

// sleep for 1 second
sleep(3000);

// maximize browser window
driver.manage().window().maximize();

}

@test(priority = 1, groups = {"PositiveTests", "NegativeTests"})
public void positiveLoginTest() {
    System.out.println("Starting login page");  
2
    // open test page
    driver.get("http://the-internet.herokuapp.com/login");
}

@Parameters({ "username", "password", "expectedMessage" })
public void negativeLoginTest(String username, String password, String expectedMessage) {
    WebDriver driver = getDriver();
    driver.get("http://the-internet.herokuapp.com/login");
    driver.findElement(By.name("username")).sendKeys(username);
    driver.findElement(By.name("password")).sendKeys(password);
    driver.findElement(By.className("button")).click();
    String actualMessage = driver.findElement(By.tagName("h3")).getText();
    assertEquals(expectedMessage, actualMessage);
}

// open test page
String url = "http://the-internet.herokuapp.com/login";
System.out.println(url);
System.out.println("Page is loaded");


```

Code see @BeforeMethod, then it will run below code, which is execute the driver first, then run GUI.

But the problem for PostiveTest is it's group, so it can't find it

8. Fix issue by adding `@BeforeMethod(alwaysRun=true)`

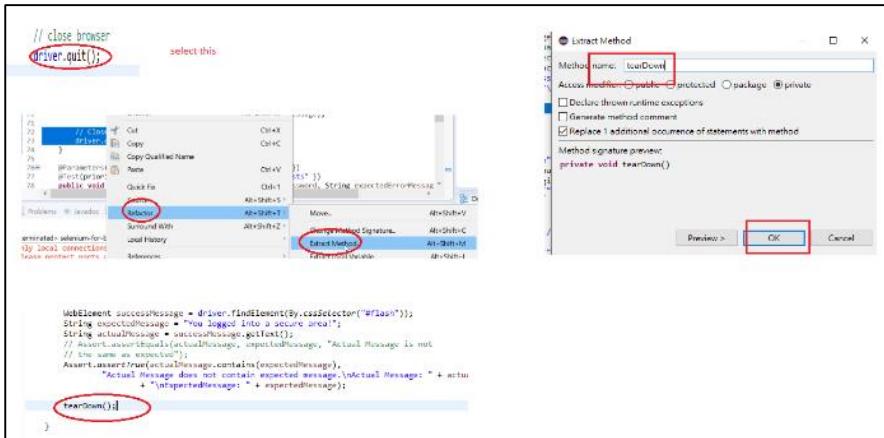
**alwaysRun** For before methods (`beforeSuite`, `beforeTest`, `beforeTestClass` and `beforeTestMethod`, but not `beforeGroups`): If set to true, this configuration method will be run **regardless of what groups it belongs to**.  
For after methods (`afterSuite`, `afterClass` ...): If set to true, this configuration method will be run even if one or more methods invoked previously failed or was skipped.  
**dependsOnGroups** The list of groups this method depends on.

If adding this it won't care about is it in group or not

```
2
3 public class LoginTests {
4
5     private WebDriver driver;
6     @BeforeMethod(alwaysRun=true)
7     private void setup() {
8
9         // create driver
10
11         System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
12         driver = new ChromeDriver();
13
14
15         // sleep for 3 second
16         sleep(3000);
17
18         // maximize browser window
19         driver.manage().window().maximize();
20
21     }
22
23     @Test(priority = 1, groups = { "positiveTests", "smokeTests" })
24     public void positiveLoginTest() {
25
26         // open browser
27         driver.get("http://www.google.com");
28
29         // enter email
30         driver.findElement(By.name("q")).sendKeys("SDET");
31
32         // click search button
33         driver.findElement(By.name("btnK")).click();
34
35         // verify result
36         String resultPageTitle = driver.getTitle();
37         assertEquals(resultPageTitle, "SDET - Google Search");
38
39     }
40 }
```

```
Starting negativeLogin Testwith tomsmithandincorrectP  
Page is opened  
=====  
NegativeTestsSuite  
Total tests run: 3, Failures: 0, Skips: 0  
=====
```

## 9. Extract driver.close to another method call tearDown



## 10. remove teardown()

delete teardown(), because after we extract method, it will have a function at below, and no matter what, it will be executable.

```
private void tearDown() {
    // close browser
    driver.quit();
}
```

Move `private void tearDown()` to below of the test, below `sleep()`

## 11. Add @AfterMethod

`@AfterMethod(alwaysRun=true)`

```
121 }
122
123@ private void sleep(long m) {
124     try {
125         Thread.sleep(m);
126     } catch (InterruptedException e) {
127         // TODO Auto-generated catch block
128         e.printStackTrace();
129     }
130 }
131
132@ @AfterMethod(alwaysRun=true)
133 private void tearDown() {
134     // close browser
135     driver.quit();
136 }
137
138 }
```

Starting negativeLogin Testwith incorrectusernameandSuperSecretPassword:  
Page is opened  
Starting ChromeDriver 79.0.3945.16 (93fcc21110c10dbbd49bbff8f472335360e31d05-refs/branch-head  
Only local connections are allowed.  
Please protect ports used by ChromeDriver and related test frameworks to prevent access by ma  
[1577577542.465][WARNING]: Timed out connecting to Chrome, retrying...  
12月 29, 2019 7:59:04 上午 org.openqa.selenium.remote.ProtocolHandshake createSession  
INFO: Detected dialect: W3C  
[1577577549.481][WARNING]: Timed out connecting to Chrome, retrying...  
Starting negativeLogin Testwith tomsmithandincorrectPassword!  
Page is opened  
|  
=====  
NegativeTestsSuite  
Total tests run: 3, Failures: 0, Skips: 0  
=====

## Test Suite4 Adding Method for browser type

Description: This Test is to add method for determine browser type, using chrome or firefox

**File used:**

LoginTests.java:

AllTest.xml: LoginTests.xml

Folder name: Case5-3\_browser\_switch

### 1. Adding parameter for setup()

```
@Parameters({ "browser" })
```

Also add string in setup parameter

```
private WebDriver driver;  
  
@Parameters({ "browser" })  
@BeforeMethod(alwaysRun=true)  
private void setUp(String browser) {  
    // create driver
```

### 2. Add switch loop for browser option

```
switch(browser name){  
case  
}
```

As below:

```
11 import org.testng.annotations.Optional;  
12 import org.testng.annotations.Parameters;  
13 import org.testng.annotations.Test;  
14  
15 public class LoginTests {  
16     private WebDriver driver;  
17  
18     @Parameters({ "browser" })  
19     @BeforeMethod(alwaysRun = true)  
20     private void setup(String browser) {  
21         // create driver  
22         switch (browser) {  
23             case "chrome":  
24                 System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");  
25                 driver = new ChromeDriver();  
26                 break;  
27             case "firefox":  
28                 System.setProperty("webdriver.gecko.driver", "src/main/resources/geckodriver.exe");  
29                 driver = new FirefoxDriver();  
30                 break;  
31             default:  
32                 System.out.println("Do not know to start " + browser + ", starting chrome browser");  
33                 System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");  
34                 driver = new ChromeDriver();  
35                 break;  
36             }  
37         }  
38     }
```

### 3. Copy driver (chrome) and paste inside switch, as below

```
System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");  
  
driver = new ChromeDriver();
```

#### 4. Modify LoginTests.xml to add browser parameter

Add this below PostiveTest

```
<parameter name="username" value="incorrectUsername" />
```

And changed to

```
<parameter name="Browser" value="chrome" />
```

Do for both positiveTests and NegativeTests

```
<suite name="NegativeTestsSuite" verbose="1">
    <test name="PositiveTests">
        <parameter name="Browser" value="chrome" />
        <groups>
            <run>
                <include name="positiveTests"></include>
            </run>
        </groups>
        <classes>
            <class name="com.herokuapp.theinternet.LoginTests" />
        </classes>
    </test>

    <test name="NegativeUsernameTest">
        <parameter name="browser" value="chrome" />
        <parameter name="username" value="incorrectUsername" />
        <parameter name="password" value="SuperSecretPassword!" />
        <parameter name="expectedMessage" value="Your username is invalid!" />
        <classes>
            <class name="com.herokuapp.theinternet.LoginTests">
                <methods>
                    <include name="negativeLoginTest" />
                </methods>
            </class>
        </classes>
    </test>

    <test name="NegativePasswordTest">
        <parameter name="browser" value="firefox" />
        <parameter name="username" value="tomsmit" />
        <parameter name="password" value="incorrectPassword!" />
        <parameter name="expectedMessage" value="Your password is invalid!" />
        <classes>
            <class name="com.herokuapp.theinternet.LoginTests">
                <methods>
                    <include name="negativeLoginTest" />
                </methods>
            </class>
        </classes>
    </test>
</suite>
```

#### 5. Test will fail

The terminal window shows the command run and its output:

```
[GPU 22/20, Chrome_Chi
##!!! [Child][MessageChannel::SendAndWait] Error: Channel e
=====
NegativeTestsSuite
Total tests run: 3, Failures: 0, Skips: 1
Configuration Failures: 2, Skips: 0
=====
```

The JUnit test report shows the execution of the NegativeTestsSuite. It includes PositiveTests, NegativeUsernameTest, and NegativePasswordTest. The PositiveTests section shows a successful run of com.herokuapp.theinternet.LoginTests with methods setUp (0 s), positiveLoginTest (0 s), and tearDown (0.002 s). The NegativeUsernameTest section shows a successful run of com.herokuapp.theinternet.LoginTests with methods negativeLoginTest (6.02 s) and a failure message "incorrectUsername","SuperSecretPassword!","Your username is invalid!" (6.02 s). The NegativePasswordTest section shows a successful run of com.herokuapp.theinternet.LoginTests with methods negativeLoginTest (6.54 s) and a failure message "tomsmit","incorrectPassword!","Your password is invalid!" (6.54 s).

The PositiveTest “Browser” is been capital, it should be small browser

```
2
3@<suite name="NegativeTestsSuite" verbose="1">
4@  <test name="PositiveTests">
5@    <parameter name="Browser" value="chrome" />
6@    <groups>
7@      <run>
8@        <include name="positiveTests"></include>
9@      </run>
10@     </groups>
11@     <classes>
12@       <class name="com.herokuapp.theinternet.LoginTest">
13@     </classes>
14@   </test>
15@ 
16@ 
17@   <test name="NegativeUsernameTest">
18@     <parameter name="Browser" value="chrome" />
19@   </test>
```

## 6. Add @optional to browser

Add optional to browser in LoginTest.java, so it can skip capitalize.

Add

```
private void setUp(@Optional("chrome") String browser)
```

Or

```
private void setUp(@Optional("") String browser)
```

```
@Parameters({ "browser" })
@BeforeMethod(alwaysRun = true)
//private void setUp(@Optional("") String browser) {
private void setUp(@Optional("chrome") String browser) {

    // create driver
    switch (browser) {
```

## 7. Go to LoginTest.xml

We can remove this or keep it, doesn't matter

```
<parameter name="Browser" value="chrome" />
```

```
=====
NegativeTestsSuite
Total tests run: 3, Failures: 0, Skips: 0
=====
```

# Part 4: Debugging

## Toggle breakpoint

### 1. Add toggle breakpoint and run with debug

Add debug.xml

```
1 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
2
3@<suite name="Debug" verbose="1">
4
5@    <test name="PositiveLoginTest">
6        <parameter name="browser" value="chrome"></parameter>
7@        <classes>
8@            <class name="com.herokuapp.theinternet.LoginTests">
9@                <methods>
10                   <include name="positiveLoginTest"></include>
11
12               </methods>
13           </class>
14       </classes>
15   </test>
16
17 </suite>
```

### 2. Let break our test, to let it fail, so we can see fail message and debug it

So add button break, since there is no this id, it will break

```
@Test(priority = 1, groups = { "positiveTests", "smokeTests" })
public void positiveLoginTest() {
    System.out.println("Starting login page ");

    // open test page
    String url = "http://the-internet.herokuapp.com/login";
    driver.get(url);
    System.out.println("Page is opened");

    // sleep for 2 second
    // sleep(2000);

    // enter username
    WebElement username = driver.findElement(By.id("username"));
    username.sendKeys("tomsmith");
    // sleep(1000);
    // enter password
    WebElement password = driver.findElement(By.name("password"));
    password.sendKeys("SuperSecretPassword!");
    sleep(3000);
    // click login button
    WebElement loginButton = driver.findElement(By.tagName("button break"));
    logInButton.click();
    sleep(5000);

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

Since we separate the driver.quit() in different function teardown. No matter run pass or fail, it will always close the browser. If you didn't separate the teardown, once it fail, the browser will not close.

```
[157/801/58.1b1][WARNING]: timed out connecting to Chrome
Starting login page
Page is opened

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

### 3. Debug message, how to see code have problem

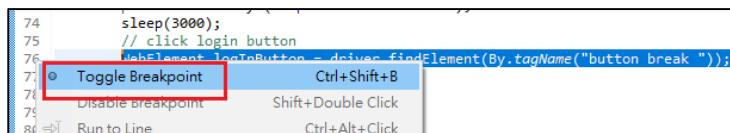
Failure Exception  
ElementException: no such element: Unable to locate element: "method": "css selector", "selector": "button\break"\n3945.79  
ror, please visit: https://www.seleniumhq.org/exceptions/no\_such\_element.html

at org.openqa.selenium.By\$ByTagName.findElement(By.java:320)  
at org.openqa.selenium.remote.RemoteWebDriver.findElement(RemoteWebDriver.java:3945.79)  
at com.herokuapp.theinternet.LoginTests.positiveLoginTest(LoginTests.java:76)  
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccess

Find your package name, and line 76 have problem

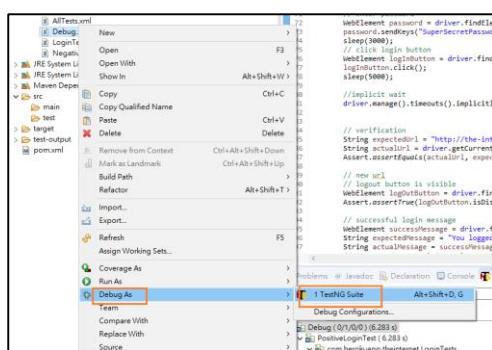
4. Sometime, it not easy to debug, we have to used debug toggle methods.

If we run debug, it will stop at the breakpoint. So let add breakpoint by below:

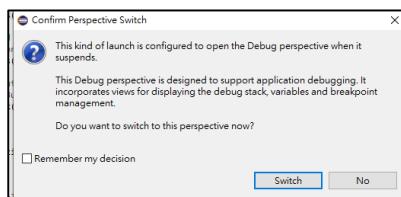


Or you can add by run>Toggle Breakpoint, or just double click it will occur a blue dot is ok.

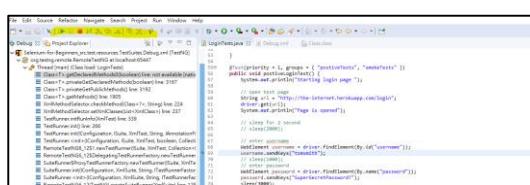
After set Toggle Break Point, then Run Debug



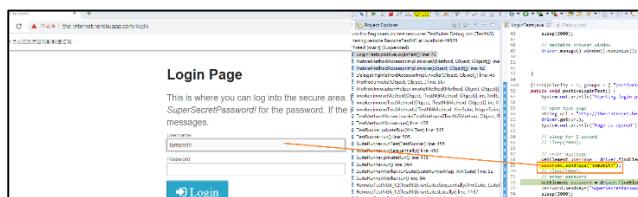
Press Switch

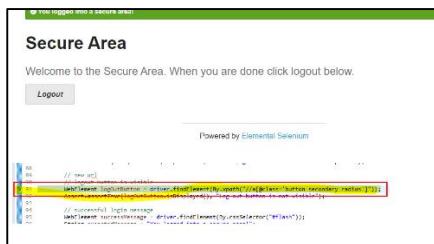


Debug will stop at toggle. This are debug option you can used



Press step over, will run one line by line





Run resume button,



It will fail

## Wait Methods

Implicit wait: wait for a certain of time wait for it to find element . is set one time and then lives for the whole like of WebDriver?

Explicitwait: is you define to wait certain of time

Run with out wait , when put toggle debug on logout code, it will stop. You run continue it will automatic fail. Explicit wait is the best and most efficient type of wait in WebDriver

Press the logout button

# Test Suite5 Implicit and Explicit wait

Description: This Test is to use different wait method

## File used:

LoginTests.java:

LoginTests.xml: Debug.xml

Folder name: debug\_implicit

## 1. Modify LoginTests.java to add Implicit wait method()

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

```
private WebDriver driver;
@Parameters({ "browser" })
@BeforeMethod(alwaysRun = true)
//private void setup(@Optional("") String browser) {
private void setup(@Optional("chrome") String browser) {

    // create driver
    switch (browser) {
        case "chrome":
            System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
            driver = new ChromeDriver();
            break;
        case "firefox":
            System.setProperty("webdriver.gecko.driver", "src/main/resources/geckodriver.exe");
            driver = new FirefoxDriver();
            break;
        default:
            System.out.println("Do not know to start " + browser + ", starting chrome browser");
            System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
            driver = new ChromeDriver();
            break;
    }
    // sleep for 3 second
    sleep(3000);
    // maximize browser window
    driver.manage().window().maximize();
    //implicit method for 10second
    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

Run debug, as previous step (press logout, then press debug, it will wait 10 second before fail)

## 2. Using Explicit way Modify LoginTests.java

Comment implicit on `private void setUp`

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

Add this for explicit way:

```
WebElement logInButton = driver.findElement(By.tagName("button"));
wait.until(ExpectedConditions.elementToBeClickable(logInButton));
logInButton.click();
```

```

    @Test(priority = 1, groups = { "positiveTests", "smokeTests" })
    public void positiveLoginTest() {
        System.out.println("Starting login page ");

        // open test page
        String url = "http://the-internet.herokuapp.com/login";
        driver.get(url);
        System.out.println("Page is opened");

        // sleep for 2 second
        // sleep(2000);

        // enter username
        WebElement username = driver.findElement(By.id("username"));
        username.sendKeys("tomsmith");
        // sleep(1000);
        // enter password
        WebElement password = driver.findElement(By.name("password"));
        password.sendKeys("SuperSecretPassword!");
        WebDriverWait wait = new WebDriverWait(driver,10);

        //sleep(3000);
        // click login button
        WebElement loginButton = driver.findElement(By.tagName("button"));
        wait.until(ExpectedConditions.elementToBeClickable(loginButton));
        loginButton.click();
        sleep(5000);
        //implicit wait
    }

```

## Test Suite6 ElementNotFoundException

### Description:

#### File used:

ExceptionsTests.java.java:

LoginTests.xml: ExceptionsTests.java.xml

Folder name: exception-wait-invisible

Test url: [http://the-internet.herokuapp.com/dynamic\\_loading/1](http://the-internet.herokuapp.com/dynamic_loading/1)

In this test if you press button, we have to wait for about 10 second wit will pop Hello worlds! The Hello world is hidden. So we have to add wait this method.

1. Create a new class name ExceptionsTests.java

Copy the code from LoginTests.

Delete Positive and negative function

2. Modify ExceptionsTests.java

Add **public void** notVisibleTest()

Add driver

**driver.get("http://the-internet.herokuapp.com/dynamic\_loading/1");**

### 3. create button locator, We will used id "start"

We can check using console, used this command `$x("//div[@id='start']/button")`

```
✖ Failed to load resource: the server responded with a status of 404 (Not Found)
> $x("//div[@id='start']/button")
< [button]
  > 0: button
    length: 1
    > __proto__: Array(0)
```

look for div element with id "start", will display button. Makes ure no other related id used it.

```
WebElement startButton=driver.findElement(By.xpath("//div[@id='start']/button"));
```

### 4. Find element

```
WebElement finishElement =driver.findElement(By.id("finish"));
```

```
String finishText=finishElement.getText();
```

```
<br>
▼<div id="start">
  <button>Start</button> == $0
</div>
▼<div id="finish" style="display:none">
  <h4>Hello World!</h4>
</div>
</div>
</div>
```

### 5. compare using Assert

```
Assert.assertTrue(finishText.contains("Hello World!));
```

```
@Test
public void notVisibleTest() {
    //open the page http://the-internet.herokuapp.com/dynamic_loading/1
    driver.get("http://the-internet.herokuapp.com/dynamic_loading/1");
    //Find locator for start Button and click on it
    WebElement startButton=driver.findElement(By.xpath("//div[@id='start']/button"));
    startButton.click();

    //Get Finished element text
    WebElement finishElement =driver.findElement(By.id("finish"));
    String finishText=finishElement.getText();

    //compare actual finish element text with expected "Hello World!" using NG Assert class
    Assert.assertTrue(finishText.contains("Hello World!"));

}
```

### 6. create new Exception.xml

```
>LoginTests.java   Debug.xml   ExceptionsTests.java   ExceptionsTests.xml
1  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
2
3  <suite name="Exception Tests Suite" verbose="1">
4
5    <test name="NotVisibleTest">
6      <parameter name="browser" value="chrome"></parameter>
7      <classes>
8        <class name="com.herokuapp.theinternet.ExceptionsTests">
9          <methods>
10            <include name="notVisibleTest"></include>
11
12            </methods>
13        </class>
14      </classes>
15    </test>
16
17 </suite>
```

## 7. Run test suite

It will Fail

The screenshot shows the Eclipse IDE's Console tab. It displays the output of a Selenium test run. The log shows several warning messages about timing out connecting to Chrome and retrying. It then indicates that the 'INFO: Detected dialect: W3C'. Below this, it says 'Exception Tests Suite' and provides a summary: 'Total tests run: 1, Failures: 1, Skips: 0'.

The screenshot shows the Eclipse IDE's Error view. It highlights a 'Failure Exception' with the message 'java.lang.AssertionError: expected [true] but found [false]' and the stack trace 'at org.testng.Assert.fail(Assert.java:96)'.

## 8. Debug it

The false message as below, so we can see the problem

```
Assert.assertTrue(finishText.contains("Hello World!"), "Finish Text:"+  
finishText);
```

It's empty so print nothing

The screenshot shows the Eclipse IDE's Error view again, with the same assertion failure message. Below it, a screenshot of a web browser shows a page with a 'start' button inside a 'div' element with 'display: none;' style. A progress bar at the bottom of the browser window is labeled 'Loading...'. This visualizes why the assertion failed, as the element was not visible.

So we have to wait for a while, it will display. In this case we can't used implicit wait

Add this code, and wait 10 second

```
WebDriverWait wait = new WebDriverWait(driver, 10);  
wait.until(ExpectedConditions.visibilityOf(finishElement));
```

The screenshot shows the Eclipse IDE's code editor with the Java test script. A blue selection bar highlights the line 'startButton.click();'. Below it, another line of code is highlighted with yellow: 'wait.until(ExpectedConditions.visibilityOf(finishElement));'. The rest of the code includes opening a URL, finding the start button by XPath, and getting the finish text for comparison.

## 9. Run again

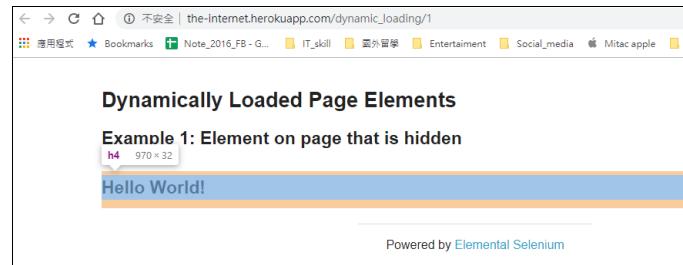
```
[1577852345.825][WARNING]: Timed out connecting to Chrome  
1/01,2020 12:19:07 下午 org.openqa.selenium.remote.ProtocolHandsh  
INFO: Detected dialect: W3C  
[1577852352.869][WARNING]: Timed out connecting to Chrome  
  
=====  
Exception Tests Suite  
Total tests run: 1, Failures: 0, Skips: 0  
=====
```

## 10. if you add click then run it, will fail

```
String finishText=finishElement.getText();  
  
//compare actual finish element text with expected "Hello World!" using NG Assert class  
Assert.assertTrue(finishText.contains("Hello World!"), "Finish Text:"+ finishText);  
  
startButton.click();  
  
}
```

```
<terminated> Selenium-for-Beginners_src.test.resources.TestSuites.ExceptionsTests.xml [T:  
[1577852911.984][WARNING]: Timed out connecting to Chrome, retrying...  
1/01,2020 12:28:34 下午 org.openqa.selenium.remote.ProtocolHandshake createSessi  
INFO: Detected dialect: W3C  
[1577852919.045][WARNING]: Timed out connecting to Chrome, retrying...  
  
=====  
Exception Tests Suite  
Total tests run: 1, Failures: 1, Skips: 0  
=====  
  
Failure Exception  
java.org.openqa.selenium.ElementNotInteractableException: element not interactable  
(Session info: chrome=79.0.3945.79)  
Build info: version: '3.141.59', revision: 'e8d578e5c9', time: '2019-11-19T00:17:00'
```

The reason is because, there is no button after we click the button. After you click will show Hello World, but will not have any button. But you try to click again will run exception error.



# Test Suite 7 Timeout exception

## Description:

### File used:

ExceptionsTests.java:

ExceptionsTests.xml:

Folder name: exception-timout

## 1. Changed `notVisibleTest()` to `timeoutTest()`

Changed 10 second to 2 seconds

```
@Test
public void timeoutTest() {
    //open the page http://the-internet.herokuapp.com/dynamic_loading/1
    driver.get("http://the-internet.herokuapp.com/dynamic_loading/1");
    //Find locator for start button and click on it
    WebElement startButton=driver.findElement(By.xpath("//div[@id='start']/button"));
    startButton.click();

    //Get Finished element text
    WebElement finishElement =driver.findElement(By.id("finish"));

    WebDriverWait wait = new WebDriverWait(driver, 2);
    wait.until(ExpectedConditions.visibilityOf(finishElement));

    String finishText=finishElement.getText();
    //compare actual finish element text with expected "Hello World!" using NG Assert class
    Assert.assertTrue(finishText.contains("Hello World!"), "Finish Text:"+ finishText);

    //startButton.click();
}
```

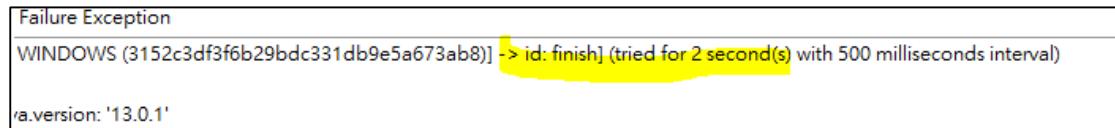
## 2. Modify Exception.xml file

Changed test name



```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Exception Tests Suite" verbose="1">
<test name="Exceptions Test">
<parameter name="browser" value="chrome"></parameter>
<classes>
<class name="com.herokuapp.theinternet.ExceptionsTests">
<methods>
<include name="timeoutTest"></include>
</methods>
</class>
</classes>
</test>
</suite>
```

## 3. Run will be fail



```
Failure Exception
WINDOWS (3152c3df3f6b29bdc331db9e5a673ab8)] -> id: finish] (tried for 2 second(s) with 500 milliseconds interval)

/a.version: '13.0.1'
```

After 2 second the element was not there. As you remember previous test when press the button need to wait from 10 second it will pop "Hello Worlds", but right now we set 2 second, so the element has not come out yet.

Element was not visible after 2 second.

#### 4. Fix this issue

We can increase the second but we don't want to do that for all the test, it might not be good.



The screenshot shows a code editor with Java code. A tooltip from the IDE's context menu is displayed over the 'catch' block. The menu items shown are: Copy Qualified Name, Paste, Quick Fix, Source, Refactor, Surround With (highlighted with a red box), Try/catch Block (highlighted with a red box), Local History, References, and Declarations. The code in the editor is:

```
89 // Then get finish element text
90 WebElement finishElement = driver.findElement(By.id("finish"));
91 WebDriverWait wait = new WebDriverWait(driver, 2);
92 wait.until(ExpectedConditions.visibilityOf(finishElement));
93 String finishText = finishElement.getText();
94 Assert.assertTrue(finishText.contains("FINISH"));
95 // compare actual finish element text
96 //startButton.click();
97 }
98 }
```

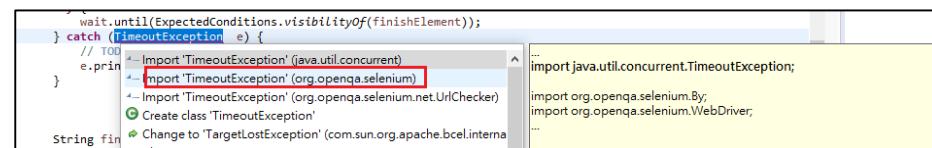
```
WebDriverWait wait = new WebDriverWait(driver, 2);
try {
    wait.until(ExpectedConditions.visibilityOf(finishElement));
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

|

This code means, it will try to run this method, if there is an exception, step inside the exception will be executed.

Changed the general exception parameter

Changed **catch (Exception e)** to **catch (TimeoutException e)**



The screenshot shows a code editor with Java code. A tooltip from the IDE's context menu is displayed over the 'catch' block. The menu items shown are: Import 'TimeoutException' (java.util.concurrent) (highlighted with a red box), Import 'TimeoutException' (org.openqa.selenium) (highlighted with a red box), Import 'TimeoutException' (org.openqa.selenium.net.UrlChecker), Create class 'TimeoutException', Change class 'TargetException' (com.sun.org.apache.bcel.internal), and ...

```
wait.until(ExpectedConditions.visibilityOf(finishElement));
} catch (TimeoutException e) {
    // TODO
    e.printStackTrace();
}
String fin
```

If exception is different than Timeout, Test will fail

Changed to this

It will print exception catch and wait for 3 second

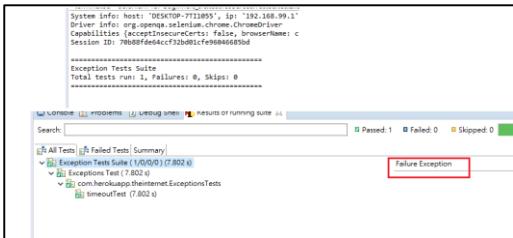


```
@Test
public void timeoutTest() {
    //open the page http://the-internet.herokuapp.com/dynamic_loading/1
    driver.get("http://the-internet.herokuapp.com/dynamic_loading/1");
    //Find locator for start Button and click on it
    WebElement startButton=driver.findElement(By.xpath("//div[@id='start']/button"));
    startButton.click();

    //Get Finished element text
    WebElement finishElement =driver.findElement(By.id("finish"));

    WebDriverWait wait = new WebDriverWait(driver, 2);
    try {
        wait.until(ExpectedConditions.visibilityOf(finishElement));
    } catch (TimeoutException exception) {
        // TODO Auto-generated catch block
        //e.printStackTrace();
        System.out.println("Exception catched:" + exception.getMessage());
        sleep(3000);
    }
}
```

## 5. Run again



```
System info: host: 'DESKTOP-7711045', ip: '192.168.99.1'
Driver info: org.openqa.selenium.chrome.ChromeDriver
Capabilities: {acceptInsecureCerts: false, browserName: chrome, ...
Session ID: 7bb0bedf64c1f3b0bcfe9084668950
...
Exception Tests Suite
Total tests run: 1, Failures: 0, Skips: 0
...
All Tests | Failed Tests | Summary
Exception Tests Suite (1/0/0/0) (7.802 s)
Exceptions Test (7.802 s)
  com.herokuapp.betheme.ExceptionsTests
    timeoutTest (7.802 s)
```

2 second of wait, and 3 second of timeout, so total wait for 5 second

## Test Suite8 No suchElementException

Description: when can't find element on page

### File used:

ExceptionsTests.java:

ExceptionsTests.xml:

Folder name: exception\_noEleemnt

[http://the-internet.herokuapp.com/dynamic\\_loading/2](http://the-internet.herokuapp.com/dynamic_loading/2)

The Hidden Hello World is not there for this link, in selenium could not find. Element is no there



```
<script>...</script>
<div class="example">
  <h3>Dynamically Loaded Page Elements</h3>
  <h4>Example 2: Element rendered after the fact</h4>
  <br>
  <div id="start">
    <button>Start</button> == $0
  </div>
  <br>
</div>
<div id="page-footer" class="row">...</div>
</body>
</html>
```

1. copy visibleTimeout() and paste to bottom of timeout name noSuchElementTest

```

3@  @Test
4  public void noSuchElementTest() {
5      // open the page http://the-internet.herokuapp.com/dynamic_loading/1
6      driver.get("http://the-internet.herokuapp.com/dynamic_loading/2");
7      // Find locator for start Button and click on it
8      WebElement startButton = driver.findElement(By.xpath("//div[@id='start']/button"));
9      startButton.click();
0
1
2      // Get Finished element text
3      WebElement finishElement = driver.findElement(By.id("finish"));
4
5      WebDriverWait wait = new WebDriverWait(driver, 10);
6      wait.until(ExpectedConditions.visibilityOf(finishElement));
7
8      String finishText = finishElement.getText();
9
10     // compare actual finish element text with expected "Hello World!" using NG
11     // Assert class
12     Assert.assertTrue(finishText.contains("Hello World!"), "Finish Text:" + finishText);
13
14 }

```

Go to Exception.xml and add the name to it

Since only three case, we will not run timeout, so we type in exclude timeoutTest

```

1 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
2
3@ <suite name="Exception Tests Suite" verbose="1"
4
5@   <test name="Exceptions Test">
6     <parameter name="browser" value="chrome"></parameter>
7@     <classes>
8@       <class name="com.herokuapp.theinternet.ExceptionsTests">
9@         <methods>
10          <exclude name="timeoutTest"></exclude>
11
12        </methods>
13      </class>
14    </classes>
15  </test>
16
17 </suite>

```

2. changed the priority as below

```

@Test(priority = 1)
public void notVisibleTest() {
    // open the page http://the-
}

@Test(priority = 2)
public void timeoutTest() {
    // ...
}

@Test(priority = 3)
public void noSuchElementTest()
    // open the page http://the-

```

3. Run it and will Fail



4. Debug and fix issue

We can't get this element `driver.findElement(By.id("finish"));`

We have to move the code

```

@Test(priority = 3)
public void noSuchElementTest() {
    // open the page http://the-internet.herokuapp.com/dynamic_loading/1
    driver.get("http://the-internet.herokuapp.com/dynamic_loading/2");
    // Find locator for start Button and click on it
    WebElement startButton = driver.findElement(By.xpath("//div[@id='start']/button"));
    startButton.click();

    // Get Finished element text
    WebElement finishElement = driver.findElement(By.id("finish"));

    WebDriverWait wait = new WebDriverWait(driver, 10);
    wait.until(ExpectedConditions.visibilityOf(finishElement));

    String finishText = finishElement.getText();

    // compare actual finish element text with expected "Hello World!" using NG
    // Assert class
    Assert.assertTrue(finishText.contains("Hello World!"), "Finish Text:" + finishText);
}

private void sleep(long s) {
}

```

Remove `visibilityOf(finishElement)` and changed to

```
wait.until(ExpectedConditions.presenceOfElementLocated(By.id("finish")));
```

```
@Test(priority = 3)
public void noSuchElementTest() {
    // open the page http://the-internet.herokuapp.com/dynamic_loading/1
    driver.get("http://the-internet.herokuapp.com/dynamic_loading/2");
    // Find locator for start Button and click on it
    WebElement startButton = driver.findElement(By.xpath("//div[@id='start']/button"));
    startButton.click();

    WebDriverWait wait = new WebDriverWait(driver, 10);
    wait.until(ExpectedConditions.presenceOfElementLocated(By.id("finish")));

    // Get Finished element text
    WebElement finishElement = driver.findElement(By.id("finish"));

    String finishText = finishElement.getText();
    // compare actual finish element text with expected "Hello World!" using NG
    // Assert class
    Assert.assertTrue(finishText.contains("Hello World!"), "Finish Text:" + finishText);
}
```

We need to return locator, so we can changed like this:

```
.presenceOfElementLocated(By.id("finish")));
+ ExpectedCondition<WebElement>
| org.openqa.selenium.support.ui.ExpectedConditions.presenceOfElementLocated(By locator)
|
An expectation for checking that an element is present on the DOM of a page. This does not necessarily
mean that the element is visible.
Parameters:
locator - locator used to find the element
Returns:
the WebElement once it is located
Press 'F2' for focus
```

WebElement `finishElement=`

```
wait.until(ExpectedConditions.presenceOfElementLocated(By.id("finish")));
```

We can now remove

```
WebElement finishElement = driver.findElement(By.id("finish"));
```

```
@Test(priority = 3)
public void noSuchElementTest() {
    // open the page http://the-internet.herokuapp.com/dynamic_loading/1
    driver.get("http://the-internet.herokuapp.com/dynamic_loading/2");
    // Find locator for start Button and click on it
    WebElement startButton = driver.findElement(By.xpath("//div[@id='start']/button"));
    startButton.click();

    WebDriverWait wait = new WebDriverWait(driver, 10);
    WebElement finishElement = wait.until(ExpectedConditions.presenceOfElementLocated(By.id("finish")));

    String finishText = finishElement.getText();
    // compare actual finish element text with expected "Hello World!" using NG
    // Assert class
    Assert.assertTrue(finishText.contains("Hello World!"), "Finish Text:" + finishText);
}

private void sleep(long m) {
    try {
        Thread.sleep(m);
    } catch (InterruptedException e) {
}
```

Go back to `ExceptionTest.xml`, and include `noSuchElementTest`

```
<suite name="Exception Tests Suite" verbose="1">
  <test name="Exceptions Test">
    <parameter name="browser" value="chrome"/>
    <classes>
      <class name="com.herokuapp.theinternet.ExceptionsTests">
        <include name="noSuchElementTest"></include>
      </class>
    </classes>
  </test>
</suite>
```

5. Run it, will pass

```
=====
Exception Tests Suite
Total tests run: 1, Failures: 0, Skips: 0
=====
```

So we wait for 10 second to let id show up then compare

## 6. Another way to solve this issue

Add this

```
Assert.assertTrue(  
    wait.until(ExpectedConditions.textToBePresentInElementLocated(By.id("fi  
nish"), "Hello World!")),  
    "Could't verify expected text 'Hello World!'");
```

```
11  @Test(priority = 3)  
12  public void noSuchElementTest() {  
13      // open the page http://the-internet.herokuapp.com/dynamic_loading/1  
14      driver.get("http://the-internet.herokuapp.com/dynamic_loading/2");  
15      // Find locator for start Button and click on it  
16      WebElement startButton = driver.findElement(By.xpath("//div[@id='start']/button"));  
17      startButton.click();  
18  
19      WebDriverWait wait = new WebDriverWait(driver, 10);  
20      /*  
21       * WebElement finishElement=  
22       * wait.until(ExpectedConditions.presenceOfElementLocated(By.id("finish")));  
23       *  
24       *  
25       * String finishText = finishElement.getText();  
26       *  
27       * // compare actual finish element text with expected "Hello World!" using NG  
28       * // Assert class Assert.assertTrue(finishText.contains("Hello World!"),  
29       * "Finish Text:" + finishText);  
30       */  
31  
32      Assert.assertTrue(  
33          wait.until(ExpectedConditions.textToBePresentInElementLocated(By.id("finish"), "Hello World!")),  
34          "Could't verify expected text 'Hello World!'");  
35  
36  }  
37  
38  private void sleep(long m) {  
39}
```

Run it

```
[1577861405.804][WARNING]: Timed out connecting to Chrome,  
1月01,2020 2:50:07 下午 org.openqa.selenium.remote.ProtocolHandshak  
INFO: Detected dialect: W3C  
[1577861412.857][WARNING]: Timed out connecting to Chrome,  
=====  
Exception Tests Suite  
Total tests run: 1, Failures: 0, Skips: 0  
=====
```

## 7. Compare the two link which are different

The screenshot shows two side-by-side browser windows. Both windows have a title bar and a main content area. The left window is labeled "before click button" and the right window is labeled "after click button". Each window displays a portion of the same web page source code.

**Left Window (before click button):**

```

http://the-internet.herokuapp.com/dynamic_loading/1
before click button
<html>
  <a href="https://github.com/tourivede/the-internet"></a>
  <div id="content" class="large-12 columns">
    <script></script>
    <div class="example">
      <h3>Dynamically Loaded Page Elements</h3>
      <br>
      <div id="start">
        <button>Start</button> => #0
        <div id="finish" style="display:none"></div>
      </div>
    </div>
  </div>
</div>

```

**Right Window (after click button):**

```

http://the-internet.herokuapp.com/dynamic_loading/1
after click button
<html>
  <a href="https://github.com/tourivede/the-internet"></a>
  <div id="content" class="large-12 columns">
    <script></script>
    <div class="example">
      <h3>Example 1: Element on page that is hidden</h3>
      <br>
      <div id="loading" style="display: none;"></div>
      <div id="start" style="display: none;">
        <button>Start</button> => #0
      </div>
      <div id="finish" style="display:></div>
    </div>
  </div>
</div>

```

In the "before" state, the "finish" element has a style of "display: none;". In the "after" state, the "loading" element is removed, and the "finish" element's style is changed to "display: none;". A yellow box highlights the "display: none;" style in both the "before" and "after" code blocks.

The first link will show the finish element, but the second will not show, it will show after you press the button.

Step are like this:

The screenshot shows a Java code editor with two test cases. The code uses Selenium WebDriver and WebDriverWait to interact with the dynamic loading page.

**Test Case 1: timeoutTest()**

```

public void timeoutTest() {
  //open the page http://the-internet.herokuapp.com/dynamic_loading/1
  driver.get("http://the-internet.herokuapp.com/dynamic_loading/1");
  //find locator for start button and click on it
  WebElement startButton=driver.findElement(By.xpath("//div[@id='start']/button"));
  startButton.click();

  //Get Finished element text
  WebElement finishElement=driver.findElement(By.id("finish"));

  WebDriverWait wait = new WebDriverWait(driver, 2);
  wait.until(ExpectedConditions.visibilityOf(finishElement));
  String finishText=finishElement.getText();

  //compare actual finish element text with expected "Hello World!" using NG Assert class
  Assert.assertEquals("Hello World!", "Finish Text:" + finishText);
}

```

**Test Case 2: noSuchElementTest()**

```

public void noSuchElementTest() {
  // open the page http://the-internet.herokuapp.com/dynamic_loading/2
  driver.get("https://the-internet.herokuapp.com/dynamic_loading/2");
  //click on Start Button and switch on it
  WebElement startButton = driver.findElement(By.xpath("//div[@id='start']/button"));

  WebDriverWait wait = new WebDriverWait(driver, 10);
  wait.until(ExpectedConditions.presenceOfElementLocated(By.id("finish")));
  String finishText = finishElement.getText();

  // compare actual finish element text with expected "Hello World!" using NG Assert class
  Assert.assertEquals("Hello World!", "Finish Text:" + finishText);
}

```

A yellow box highlights the "wait.until(ExpectedConditions.visibilityOf(finishElement));" line in the first test case and the "wait.until(ExpectedConditions.presenceOfElementLocated(By.id("finish")));" line in the second test case.

## Test Suite9: Tricky Exception:

### StaleElementReferenceException

Description: The element have been remove

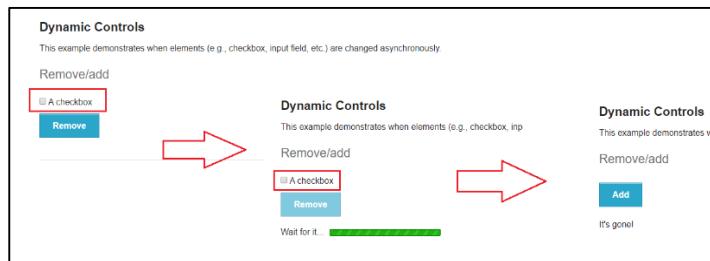
#### File used:

ExceptionsTests.java:

ExceptionsTests.xml:

Folder name: slame

Test Link: [http://the-internet.herokuapp.com/dynamic\\_controls](http://the-internet.herokuapp.com/dynamic_controls)



1. Create a method public void staleElementTest()
2. Find the correct locator to used

Go to chrome and inspect console page and type in

`$x("//button[contains(text(),'Remove')]")`

```
> $x("//button[contains(text(),'Remove'))")
<  ▼ [button]
  ▶ 0: button
    length: 1
  ▶ __proto__: Array(0)
```

Two button

```
> $x("//button")
<  ▼ (2) [button, button]
  ▶ 0: button
  ▶ 1: button
    length: 2
  ▶ __proto__: Array(0)
```

### 3. Add the code in staleElementTest

```
@Test  
public void staleElementTest() {  
  
    driver.get("http://the-internet.herokuapp.com/dynamic_controls");  
    WebElement checkbox=driver.findElement(By.id("checkbox"));  
    WebElement removeButton=driver.findElement(By.xpath("//button[contains(text(),'Remove')]"));  
    WebDriverWait wait = new WebDriverWait(driver,10);  
    removeButton.click();  
    wait.until(ExpectedConditions.invisibilityOf(checkbox));  
    Assert.assertFalse(checkbox.isDisplayed());  
}  
}
```

### 4. add the method in the testsuite XML FILE

```
<x:Debug> <x:ExceptionTests> <x:ExceptionsTests> <x:ExceptionsTestsXML>  
1 <!DOCTYPE suite SYSTEM "http://testing.org/testing-1.0.dtd">  
2  
3<!suite name="Exception Tests Suite" verbose="1">  
4  
5<test name="Exceptions Test">  
6    <parameter name="browser" value="chrome"/>  
7</test>  
8<class name="com.herokuapp.theinternet.ExceptionsTests">  
9<methods>  
10    <method>  
11        <include name="staleElementTest" />  
12    </methods>  
13</class>  
14</classes>  
15</test>  
16</testsuite>  
17
```

### 5. Run it and fail

```
Failure Exception  
num.StaleElementReferenceException: stale element reference: element is not attached to the page document  
rome=79.0.3945.79)  
on on this error, please visit: https://www.seleniumhq.org/exceptions/stale_element_reference.html  
n: '3.141.59', revision: 'e82be7d358', time: '2018-11-14T08:17:03'  
: 'DESKTOP-7TI105', ip: '192.168.99.1', os.name: 'Windows 10', os.arch: 'amd64', os.version: '10.0', java.version: '13.0.1'
```

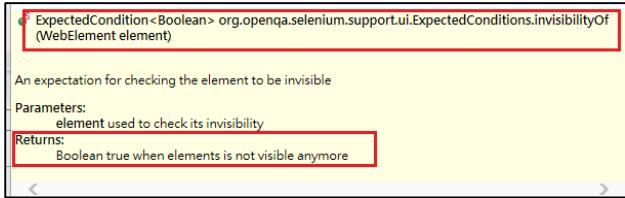
It fail on this line `Assert.assertFalse(checkbox.isDisplayed());`

Because the element(checkbox) is been remove so it can't find so it fail.

The checkbox was remove for the page, so we no longer can used WebElement checkbox.

So when we run `checkbox.isDisplayed()`, it will fail on StaleElementReferenceException error.

## 6. Fix this problem by this:



Comment out this code first

Comment out this code first

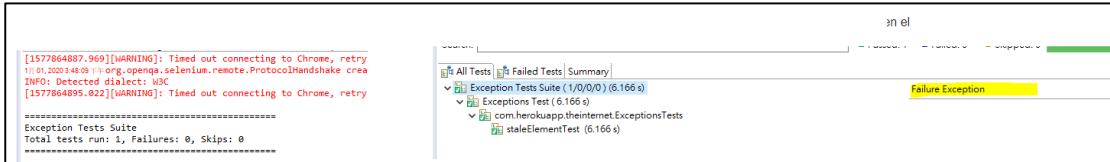
```
/*
```

```
* wait.until(ExpectedConditions.invisibilityOf(checkbox));  
* Assert.assertFalse(checkbox.isDisplayed());  
*/
```

Add below code:

```
Assert.assertTrue(wait.until(ExpectedConditions.invisibilityOf(checkbox)), "Checkbox is still visible, but  
shouldn't be");
```

Run it will pass

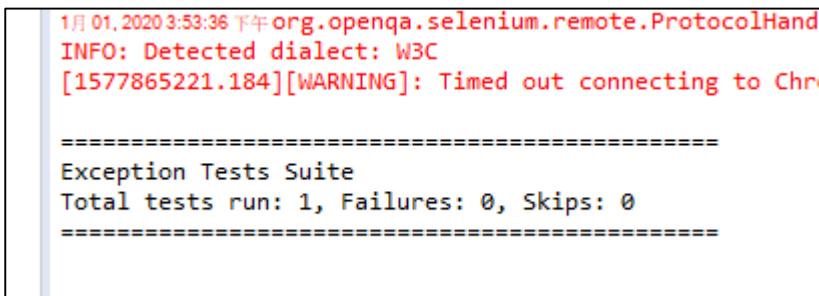


Note: not all case will pass

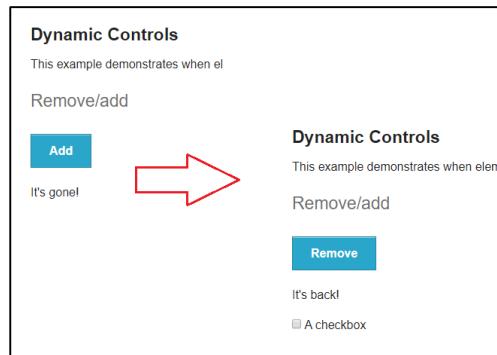
## 6. Another method

Comment the code again

```
Assert.assertTrue(wait.until(ExpectedConditions.stalenessOf(checkbox)),  
"CheckBox is still visible, but shouldn't be");
```



Click on the add and wait for the checkbox to come out



Inspect checkbox

```
> $x("//button[contains(text(),'Remove'))")
< ▶ [button]
> $x("//button[contains(text(),'Add'))")
< ▶ []
> $x("//button[contains(text(),'Add'))")
< ▶ []
> $x("//button[contains(text(),'Add'))")
< ▶ [button] ██████████
>
```

Add the code

```
Assert.assertTrue(wait.until(ExpectedConditions.stalenessOf(checkbox)),
"CheckBox is still visible, but shouldn't be");
```

WebElement

```
addButton=driver.findElement(By.xpath("//button[contains(text(),'Add'))"));
addButton.click();
wait.until(ExpectedConditions.visibilityOf(checkbox));
Assert.assertTrue(checkbox.isDisplayed(), "Checkbox is not visible,
but it shiould be ");
```

```

139@  @Test
140  public void staleElementTest() {
141
142      driver.get("http://the-internet.herokuapp.com/dynamic_controls");
143      WebElement checkbox=driver.findElement(By.id("checkbox"));
144      WebElement removeButton=driver.findElement(By.xpath("//button[contains(text(),'Remove')]"));
145      WebDriverWait wait = new WebDriverWait(driver,10);
146      removeButton.click();
147      /*
148      * wait.until(ExpectedConditions.invisibilityOf(checkbox));
149      * Assert.assertFalse(checkbox.isDisplayed());
150      */
151
152      //Assert.assertTrue(wait.until(ExpectedConditions.invisibilityOf(checkbox)), "Checkbox is still visible, but shouldn't be");
153      Assert.assertTrue(wait.until(ExpectedConditions.stalenessOf(checkbox)), "CheckBox is still visible, but shouldn't be");
154
155      WebElement addButton=driver.findElement(By.xpath("//button[contains(text(),'Add')]"));
156      addButton.click();
157      wait.until(ExpectedConditions.visibilityOf(checkbox));
158      Assert.assertTrue(checkbox.isDisplayed(), "Checkbox is not visible, but it shiould be ");
159
160  }
161
162@ private void sleep(long m) {
163     try {
164         Thread.sleep(m);
165     } catch (InterruptedException e) {
166         // TODO Auto-generated catch block

```

Will fail, most common people will face on this problem

Failure Exception
<pre>J\ org.openqa.selenium.StaleElementReferenceException: stale element reference: element is not attached to the page document (Session info: chrome=79.0.3945.79) For documentation on this error, please visit: https://www.seleniumhq.org/exceptions/stale_element_reference.html Build info: version: '3.141.59', revision: 'e82be7d358', time: '2018-11-14T08:17:03' System info: host: 'DESKTOP-7TI1055', ip: '192.168.99.1', os.name: 'Windows 10', os.arch: 'amd64', os.version: '10.0', java.version:</pre>

Because checkbox is removed from the page, new locator with same id was added to the page

Remove or comment

```
//wait.until(ExpectedConditions.visibilityOf(checkbox));
//Assert.assertTrue(checkbox.isDisplayed(), "Checkbox is not visible, but it
should be ");
```

Add this

```
WebElement checkbox2 =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("checkbox")))
;
Assert.assertTrue(checkbox2.isDisplayed(), "Checkbox is not visible, but it
should be");
```

```

@Test
public void staleElementTest() {
    driver.get("http://the-internet.herokuapp.com/dynamic_controls");
    WebElement checkbox=driver.findElement(By.id("checkbox"));
    WebElement removeButton=driver.findElement(By.xpath("//button[contains(text(),'Remove')]"));
    WebDriverWait wait = new WebDriverWait(driver,10);
    removeButton.click();
    /*
    * wait.until(ExpectedConditions.invisibilityOf(checkbox));
    * Assert.assertFalse(checkbox.isDisplayed());
    */
    //Assert.assertTrue(wait.until(ExpectedConditions.invisibilityOf(checkbox)), "Checkbox is still visible, but shouldn't be");
    Assert.assertTrue(wait.until(ExpectedConditions.stalenessOf(checkbox)), "CheckBox is still visible, but shouldn't be");

    WebElement addButton=driver.findElement(By.xpath("//button[contains(text(),'Add')]"));
    addButton.click();
    //wait.until(ExpectedConditions.visibilityOf(checkbox));
    //Assert.assertTrue(checkbox.isDisplayed(), "Chckbox is not visible, but it shiould be ");

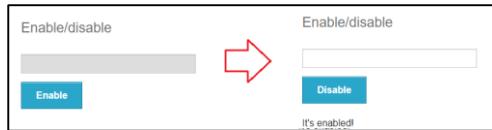
    WebElement checkbox2 = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("checkbox")));
    Assert.assertTrue(checkbox2.isDisplayed(), "Checkbox is not visible, but it should be");
}
private void sleep(long m)
```

```

[1577866740.583][WARNING]: Timed out connecting to Chrome, retr
1月01,2020 4:19:02 下午 org.openqa.selenium.remote.ProtocolHandshake cre
INFO: Detected dialect: W3C
[1577866747.638][WARNING]: Timed out connecting to Chrome, retr

=====
Exception Tests Suite
Total tests run: 1, Failures: 0, Skips: 0
=====
```

5. This is same as previous link, just



## Part 5: Summary

### Test Suite-Summarize All

#### 1. Using group:

```

2<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
3<suite name="NegativeTestsSuite" verbose="1">
4
5@<test name="NegativeTests">
6@<groups>
7@<run>
8@<include name="smokeTests" />
9
10</groups>
11</test>
12
13<classes>
14<class name="com.herokuapp.theinternet.NegativeTests" />
15
```

```

public class NegativeTests {
    @Test(priority=1, groups = {"negativeTests", "smokeTests"})
    public void incorrectUsernameTest() {
        System.out.println("Starting incorrectUsername Test");
    }
    @Test(priority=2, groups = {"negativeTests"})
    public void incorrectPasswordTest() {
        System.out.println("Starting incorrectPassword Test");
    }
}
```

Will run incorrectUsernameTest, because it have smokeTest as group

#### 2. using Include:

Run incorrectUsernameTest, when exclude incorrectPasswordTest

```

Selenium-for-Beginners/pom.xml NegativeTests.java NegativeTests.xml
1<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
2
3<suite name="NegativeTestsSuite" verbose="1">
4
5@<test name="NegativeTests">
6@<groups> <run> <include name="smokeTests" /> </run> </groups> -->
7
8
9<classes>
10<class name="com.herokuapp.theinternet.NegativeTests">
11<methods>
12<exclude name="incorrectPasswordTest" />
13
14</methods>
15</class>
```

```

public class NegativeTests {
    @Test(priority=1, groups = {"negativeTests", "smokeTests"})
    public void incorrectUsernameTest() {
        ...
    }
    @Test(priority=2, groups = {"negativeTests"})
    public void incorrectPasswordTest() {
        ...
    }
}
```

Run incorrectPasswordTest, when include incorrectPasswordTest

### 3. using Parameter

```
negativeTest.java
public class NegativeTests {
    @Parameters({ "username", "password", "expectedMessage" })
    @Test(priority = 1, groups= { "negativeTests", "smokeTests" })
    public void NegativeLoginTest(String username, String password, String expectedErrorMessage) {
        System.out.println("Starting negative login test with "+username+" and "+password);
        // create driver

        System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
        System.setProperty("webdriver.gecko.driver", "src/main/resources/geckodriver.exe");
        WebDriver driver = new ChromeDriver();
        //WebDriver driver = new FirefoxDriver();
        // sleep for 3 second
        sleep(3000);

        // maximize browser window
        driver.manage().window().maximize();

        // open test page
        String url = "http://the-internet.herokuapp.com/login";
        driver.get(url);
        System.out.println("Page is opened");

        // sleep for 2 second
        sleep(2000);

        // enter username
        WebElement usernameElement = driver.findElement(By.id("username"));
        usernameElement.sendKeys("incorrectUsername");
        usernameElement.sendKeys(username);
        sleep(1000);
        // enter password
        WebElement passwordElement = driver.findElement(By.name("password"));
        passwordElement.sendKeys("SuperSecretPassword!");
        passwordElement.sendKeys(password);
        sleep(3000);
        // click login button
    }
}
```

```
NegativeTest.xml
<suite name="NegativeTestsSuite" verbose="1">
    <test name="NegativeUsernameTests">
        <parameter name="username" value="incorrectUsername" />
        <parameter name="password" value="SuperSecretPassword!" />
        <parameter name="expectedMessage" value="Your username is invalid!" />
        <classes>
            <class name="com.herokuapp.theInternet.NegativeTests">
                </class>
        </classes>
    </test>

    <test name="NegativePasswordTests">
        <parameter name="username" value="correctUsername" />
        <parameter name="password" value="IncorrectPassword!" />
        <parameter name="expectedMessage" value="Your password is invalid!" />
        <classes>
            <class name="com.herokuapp.theInternet.NegativeTests">
                </class>
        </classes>
    </test>

```

Due to NegativeTests is the class, it will read the file

### 4. Using All method all in one: (include, group, and parameter)

```
LoginTests.xml
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="PositiveTestsSuite" verbose="1">
    <test name="PositiveTests" groups="positiveTests">
        <groups>
            <run>
                <include name="positiveTests"/>
            </run>
        </groups>
        <classes>
            <class name="com.herokuapp.theInternet.LoginTests" />
        </classes>
    </test>
    <test name="NegativeUsernameTest">
        <parameter name="username" value="incorrectUsername" />
        <parameter name="password" value="SuperSecretPassword!" />
        <parameter name="expectedMessage" value="Your username is invalid!" />
        <classes>
            <class name="com.herokuapp.theInternet.LoginTests" />
        </classes>
    </test>
    <test name="NegativePasswordTest">
        <parameter name="username" value="correctUsername" />
        <parameter name="password" value="IncorrectPassword!" />
        <parameter name="expectedMessage" value="Your password is invalid!" />
        <classes>
            <class name="com.herokuapp.theInternet.LoginTests" />
        </classes>
    </test>

```

```
LoginTest.java
package com.herokuapp.theinternet;
import org.openqa.selenium.By;
public class LoginTests {
    @Test(priority = 1, groups= { "positiveTests", "smokeTests" })
    public void PositiveLoginTest() {
        // create driver
        System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
        System.setProperty("webdriver.gecko.driver", "src/main/resources/geckodriver.exe");
        WebDriver driver = new ChromeDriver();
        // sleep for 3 second
        sleep(3000);

        // maximize browser window
        driver.manage().window().maximize();

        // open test page
        String url = "http://the-internet.herokuapp.com/login";
    }
}
```

## Part 6: Reference

### 1. Update Marven Project

