# 内容


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 suchElelementException ..... 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

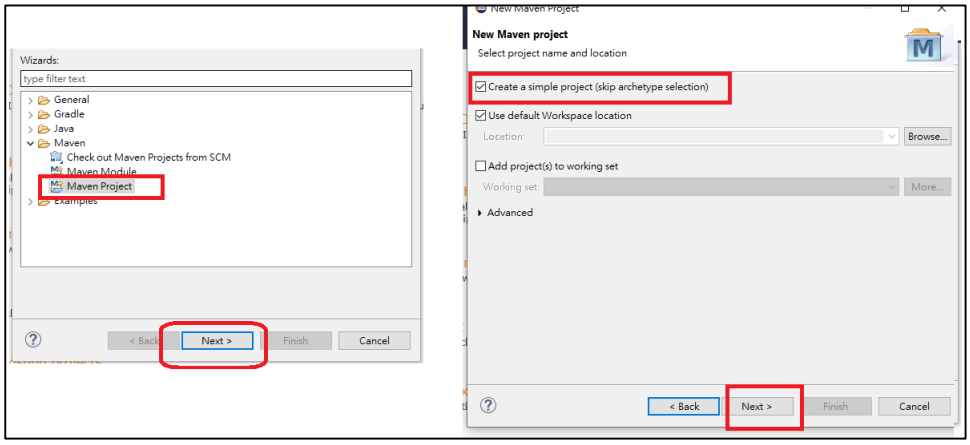
```
java -version // this will check your jre version
javac -version // this will check your java compiler version if you installed the jdk
```

## 2. Eclipse Setting

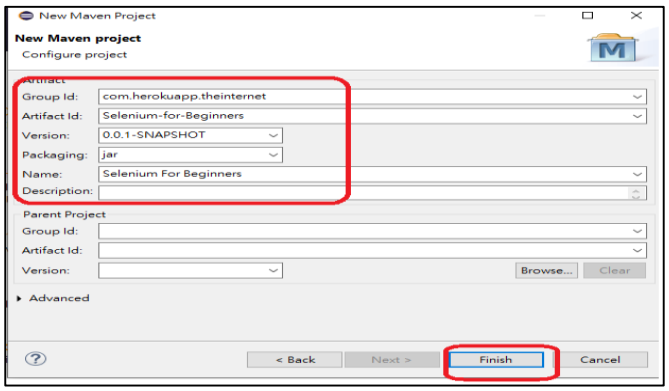
Create a New Project

1.) Create new project (File->new->new project)

2.) Select Maven-> Maven Project ->next

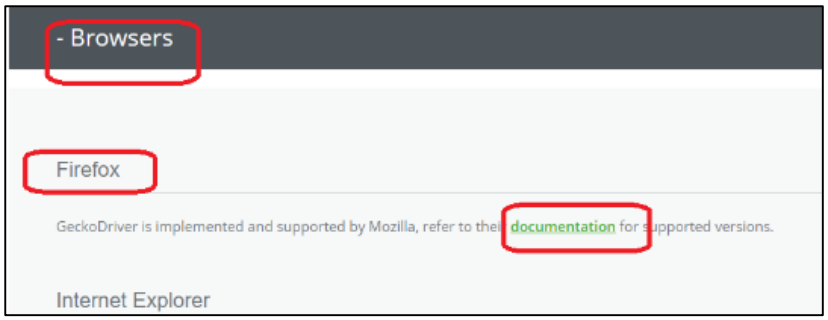



3.) Add the Group ID

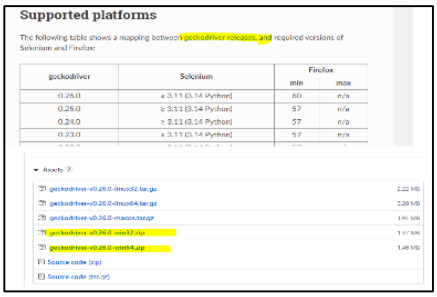


4.) Download Selenium Driver

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



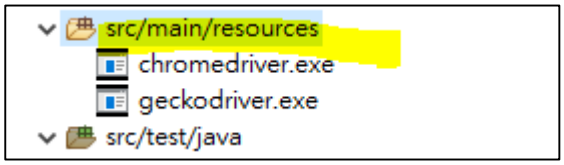

Go to documentation to download chrome and firefox driver.

| **Firefox:** | **Chrome:** |
| --- | --- |
|  |  |



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

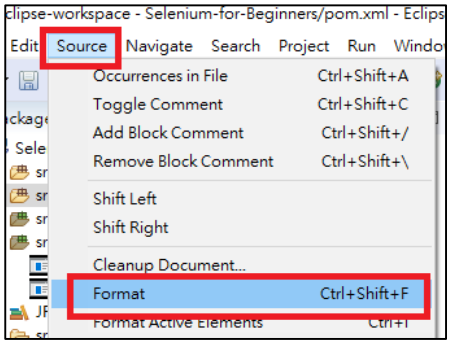
Link: https://maven.apache.org/plugins/maven-compiler-plugin/examples/set-compiler-source-and-target.html

2. Copy the xml text and add into pol.xml

Copy from <built> to </build>

```
<build>
  [...]
  <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 pod.xml into correct java version

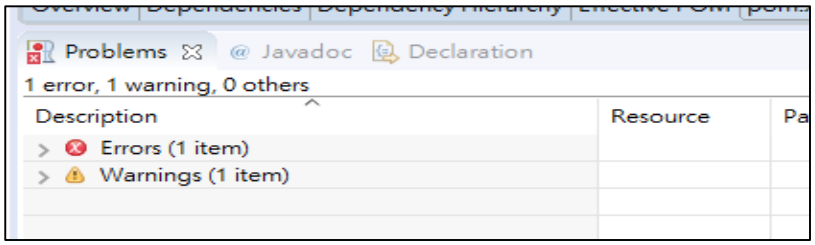
I used java 13.0, so I will change to 13. (we can use java --version to see java version)


Modify as:

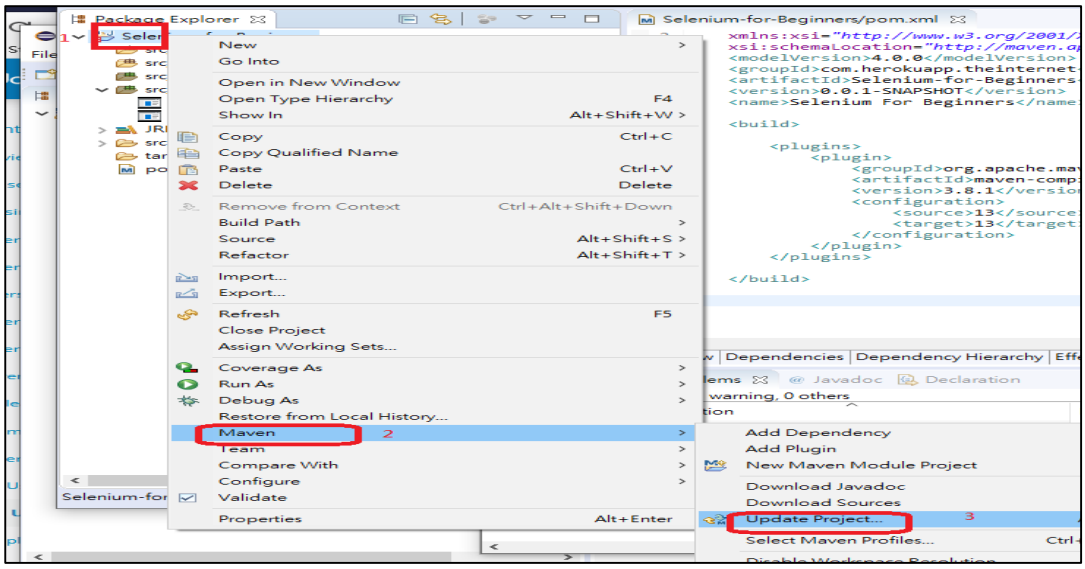
```
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.1</version>
    <configuration>
      <source>13</source>
      <target>13</target>
    </configuration>
  </plugin>
</plugins>
```

4. Make sure problem will not have warning or error

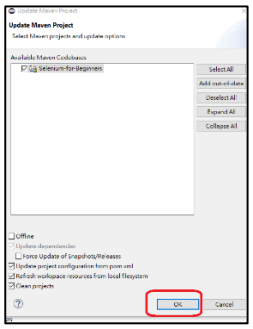
After save it button will occur error

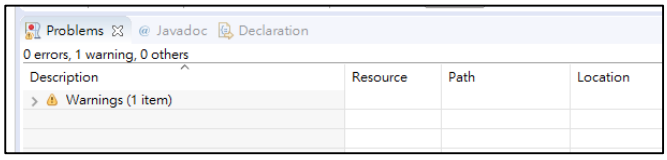



Fixed by using this: right click and press Marven>update project

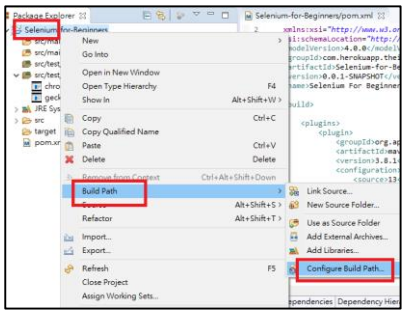


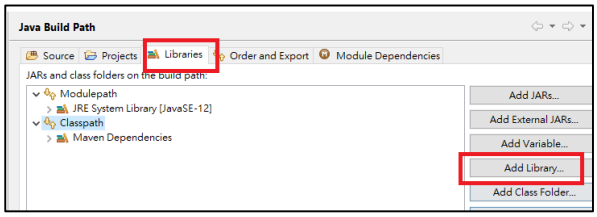

Press ok

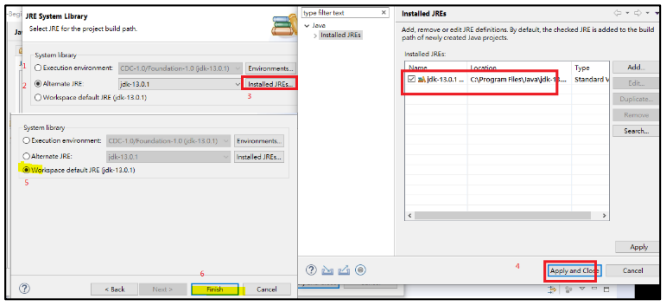


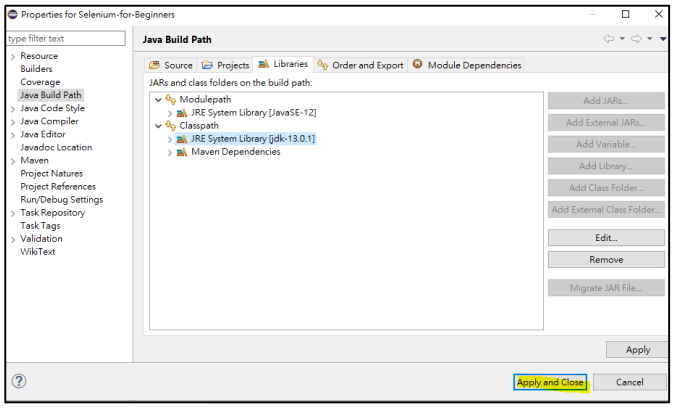


5. If {problem still occur warning, please do below step. (if no warning you can skip).
Your JDK might not match your Eclipse, need to do some setting

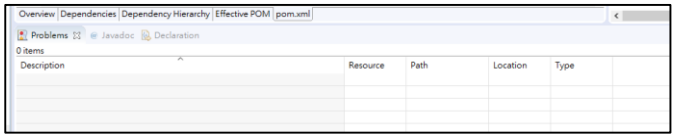



And go to update maven project again (just like you did in step 9). No error or warning as below

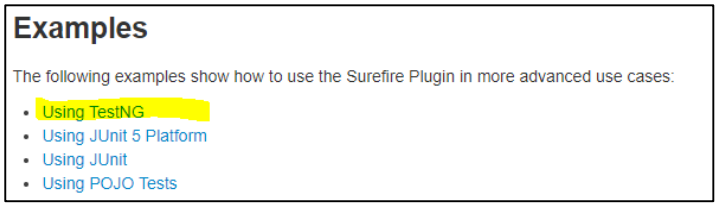


Maven Surefire Plugin

1.) Create Maven Surefire Plugin file

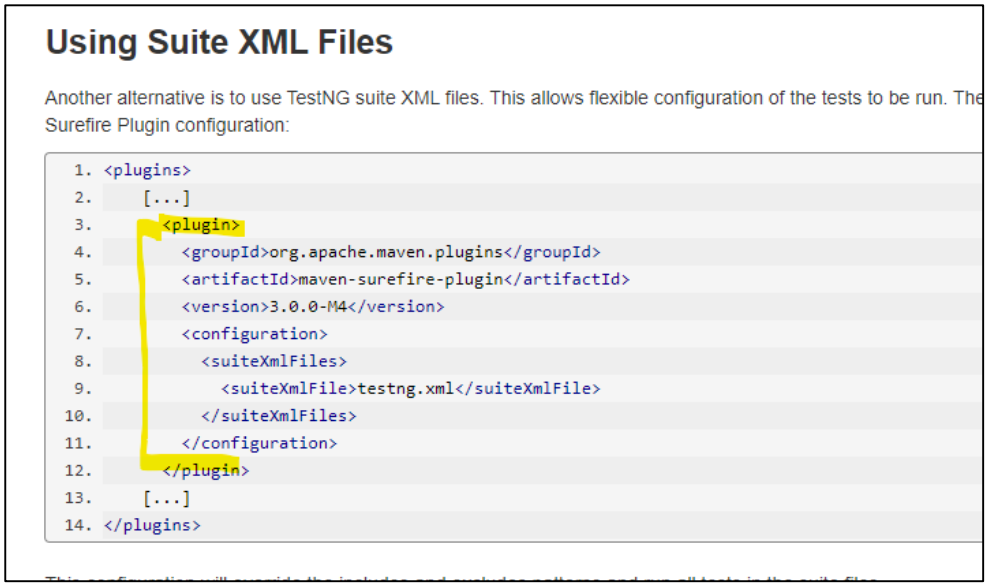
1. Go to Maven Surefire link as below

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




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

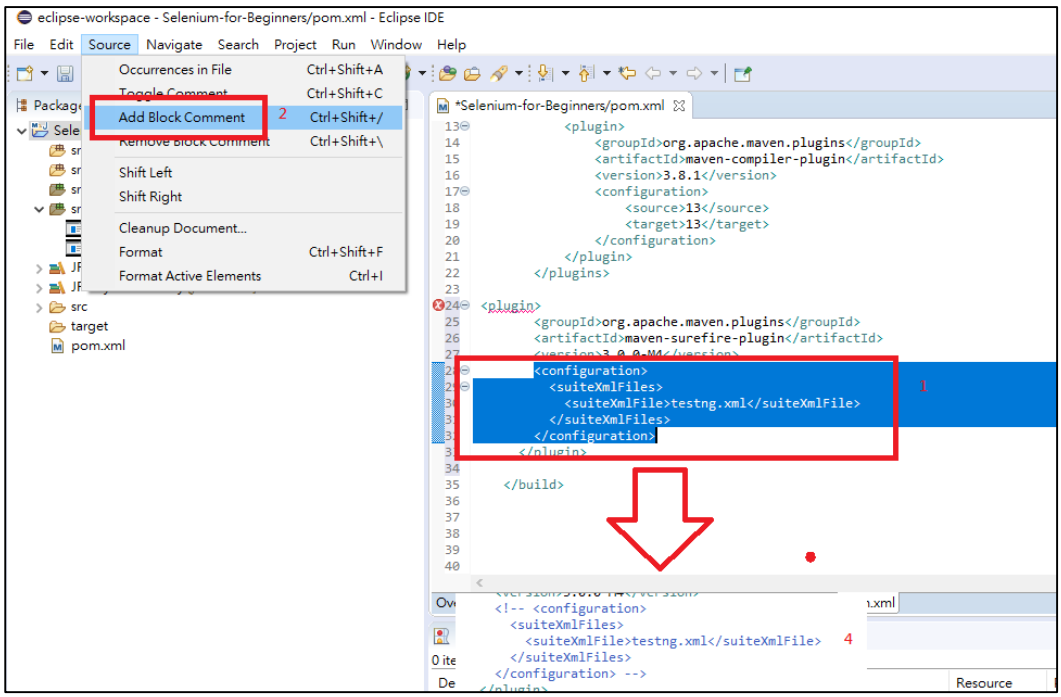
Press “Using Testing” from above link, and copy “Using Suite XML Files ‘s plugin” and paste into pom.xml




Copy to pom.xml as below

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

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



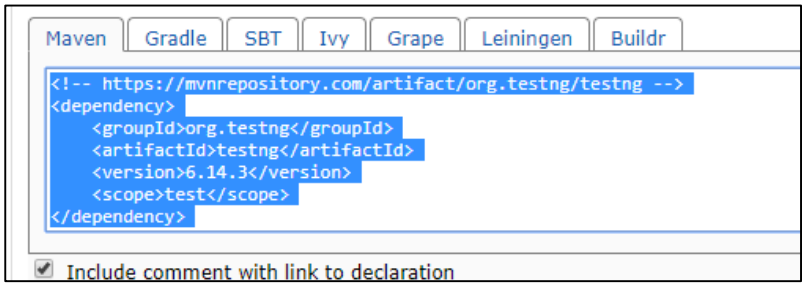

Press format to make your code nicer. Update maven project again

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

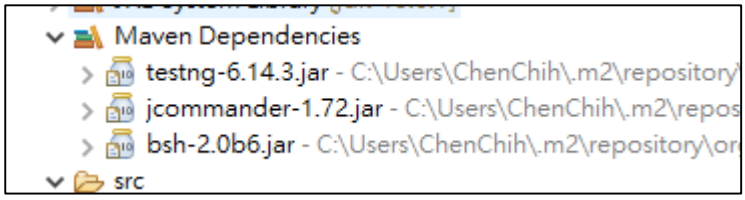
Copy below



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

```
<artifactId>Selenium-for-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 Marven 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>
```

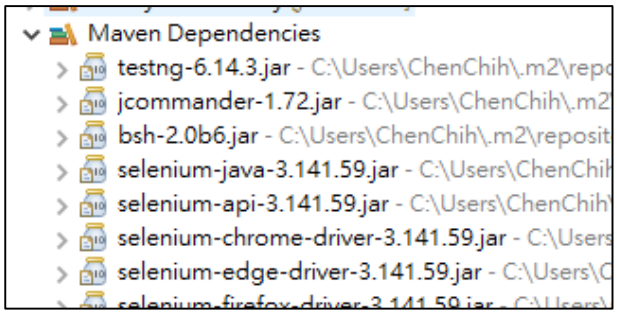
3. Copy to pom.xml and save

```
<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

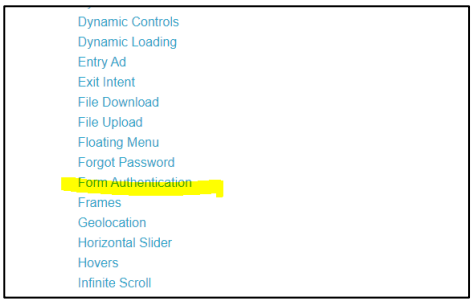

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.testng/testng -->
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>6.14.3</version>

    </dependency>
    <!-- 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>


</dependencies>

<build>

    <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>

        <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>
```

# Part 2: Create Test Plan

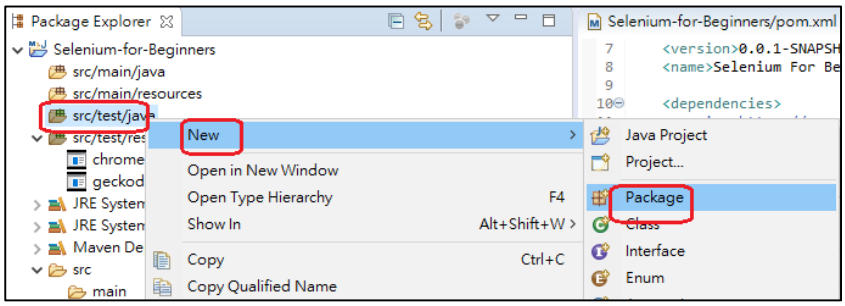
## Test Plan1-Create Open Web Browser

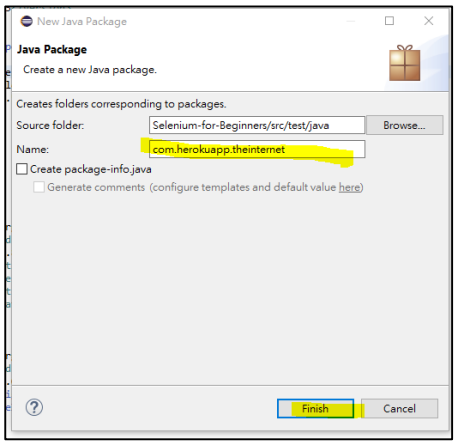
1. Create Test Case, used which type of link you want to test

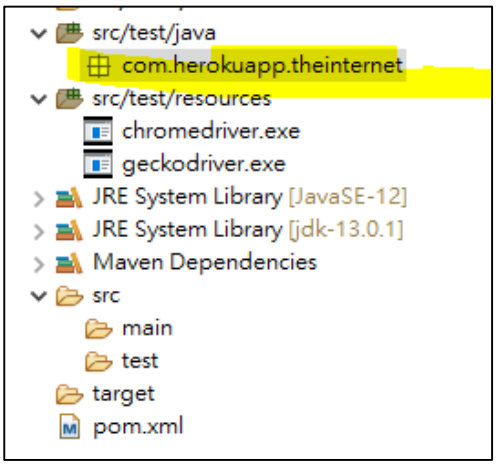
Test URL: http://the-internet.herokuapp.com/



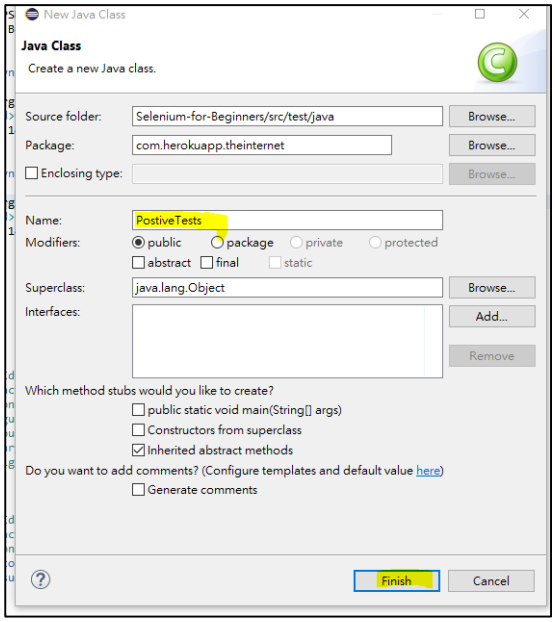

2. Create package

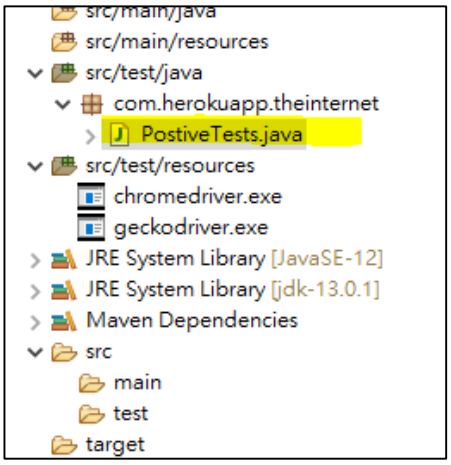


3. Create class name PsostiveTest.java

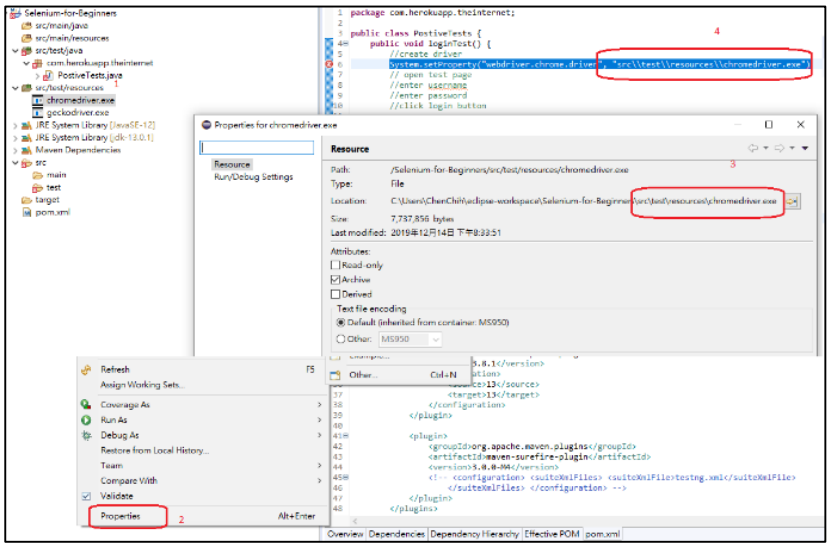


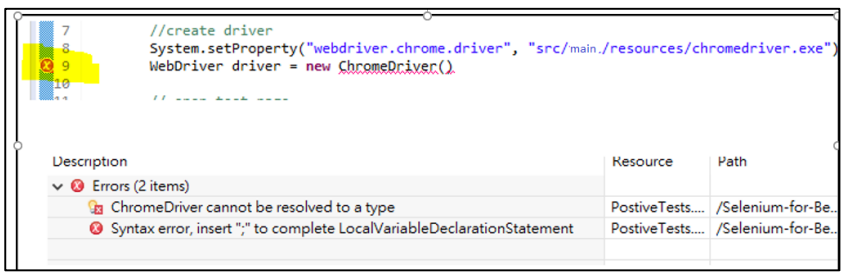

4. Import driver in the file PostiveTets.java

Syntax: System.setProperty(key, value)

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



*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;
```

```
package com.herokuapp.theinternet;

import org.openqa.selenium.WebDriver;

public class PostiveTests {
    public void loginTest() {
        //create driver
        System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe"
        WebDriver driver = new ChromeDriver();

        // open test page
        //enter username
        //enter password
        //click login button

        //verification
        //new url
        //logout button is visible
        //successful login message
```

We can use Eclipse method to import it:

```
public class PostiveTests {
    public void loginTest() {
        //create driver
        System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
        WebDriver driver = new ChromeDriver();

        // open test page
        //enter username
        //enter password
        //click login button

        //verification
        //new url
        //logout button is visi
        //successful login mess


    }
```

ChromeDriver cannot be resolved to a type

7 quick fixes available:

- Import 'ChromeDriver' (org.openqa.selenium.chrome)
- Create class 'ChromeDriver'
- Change to 'WebDriver' (org.openqa.selenium)
- Change to 'ChromeDriverInfo' (org.openqa.selenium.chrome)
- Change to 'ChromeDriverService' (org.openqa.selenium.chrome)
- Change to 'ChromeFilter' (org.openqa.selenium.remote.session)
- Fix project setup...

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

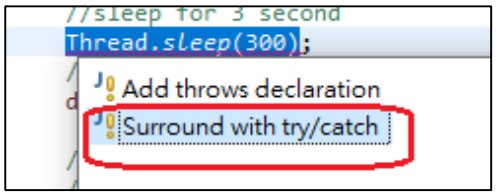
sleep 3 second

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

8. Extract sleep method

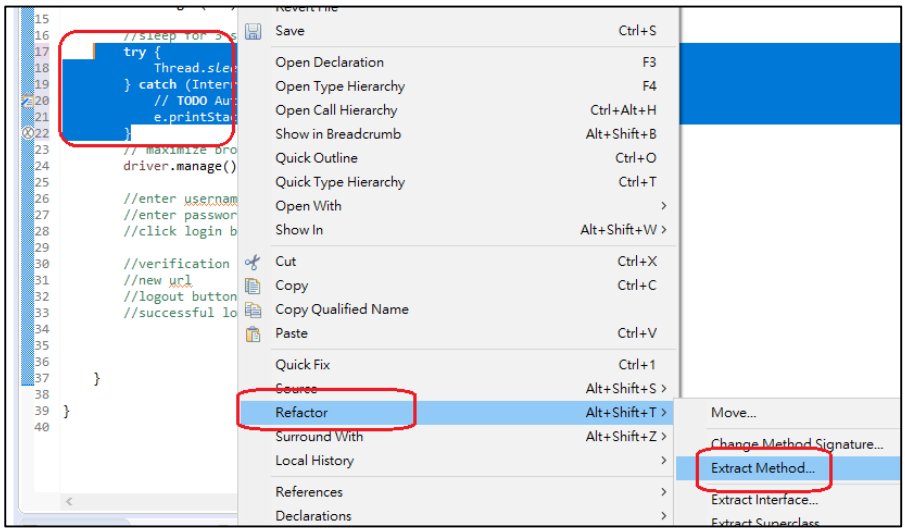
Adding sleep method from previous for better coding

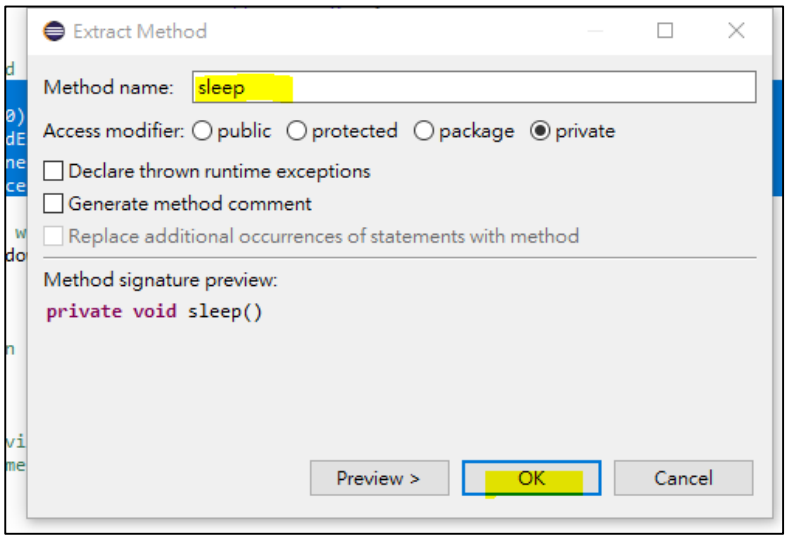
The purpose of this will let code to be more efficient, used don't have to type 3000, they can type n second.




```
//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:

```
    public void loginTest() {
        //create driver
        System.setProperty("webdriver.chrome.driver", "src/test/resources
        WebDriver driver = new ChromeDriver();

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

        //sleep for 3 second
        sleep();

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

        //enter username
        //enter password
        //click login button

        //verification
        //new url
        //logout button is visible
        //successful login message


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

9. Modify sleep()

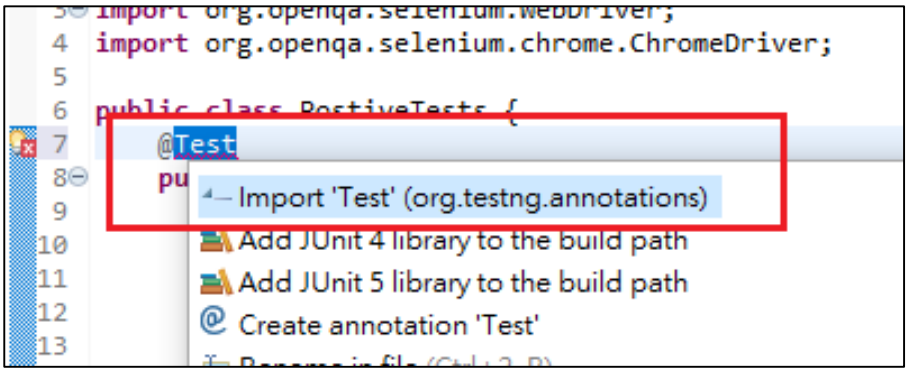
move sleep() under driver, and edit the function. Add variable to it

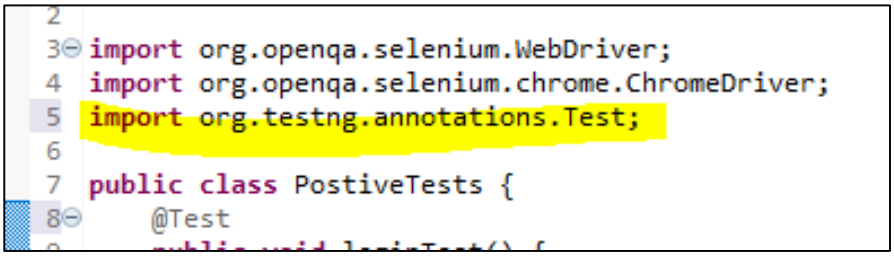
```
    private void sleep(long m) {
        try {
            Thread.sleep(m);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

}
```

```
public class PostiveTests {
    public void loginTest() {
        //create driver
        System.setProperty("webdriver.chrome.driver", "src/test/resources/chromedriver
        WebDriver driver = new ChromeDriver();

        System.out.println("Starting login page ");

        //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
        //enter password
        //click login button

        //verification
        //new url
        //logout button is visible
        //successful login message
```

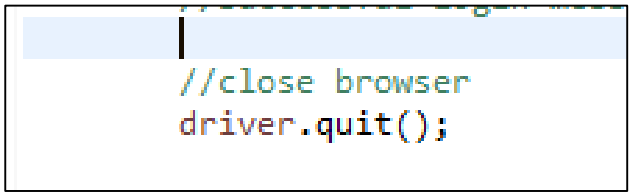
10. Add @test notation

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




11. Close Browser



# Test Plan2- Run Open Web Browser

Description: This Test is just open browser

**File used:**

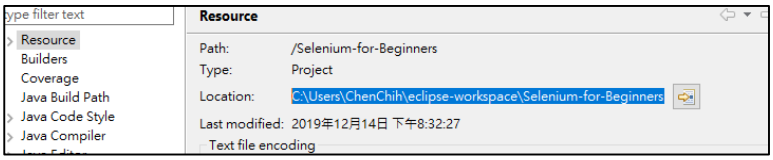
PostiveTests.java

pom.xml

Folder name: Case1-1_openWEB

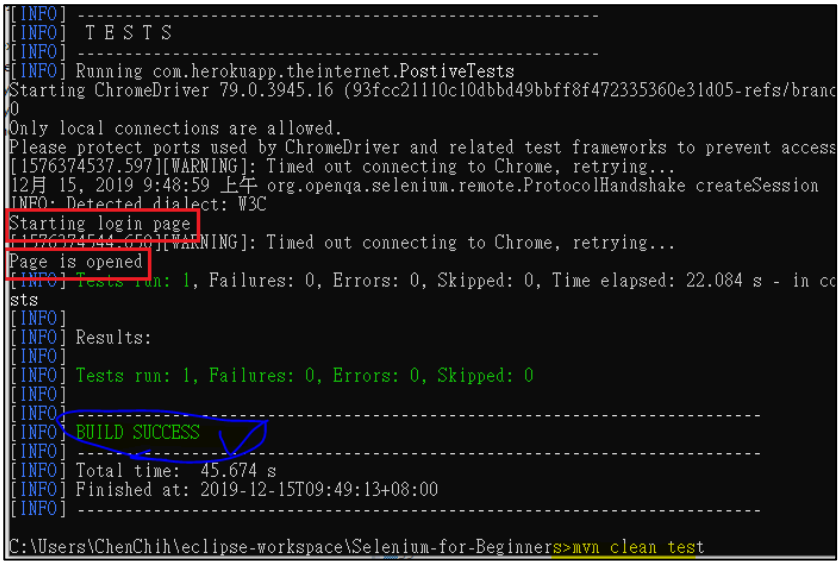
1. Check your file directory and location

Press right click>properties to find your package location.




2. Go to CMD prompt to run test

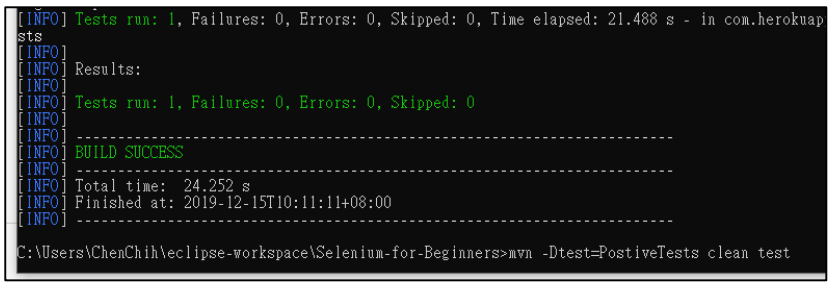
Execute: mvn clean test




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

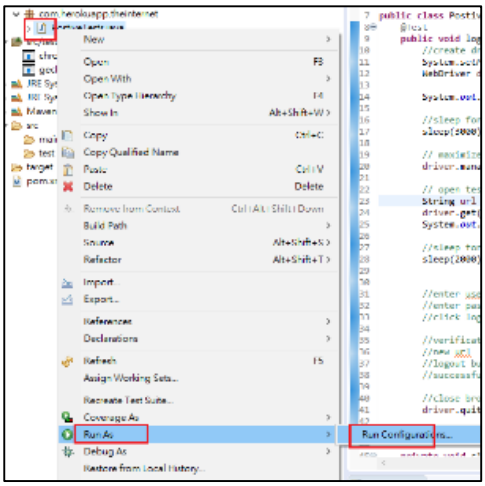
3. Add test case under prompt

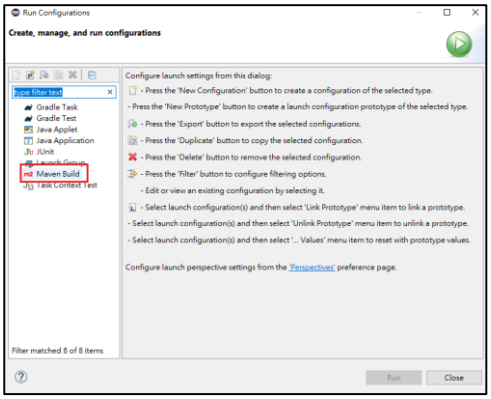
mvn -Dtest=PostiveTests clean test

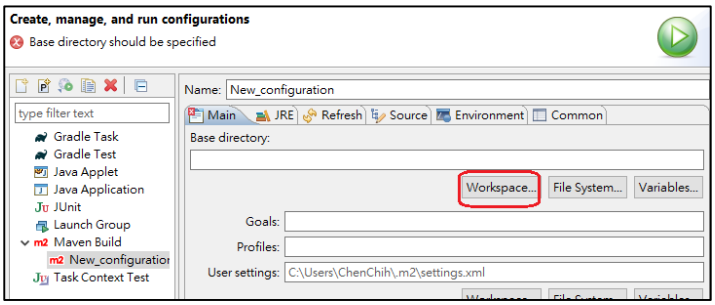


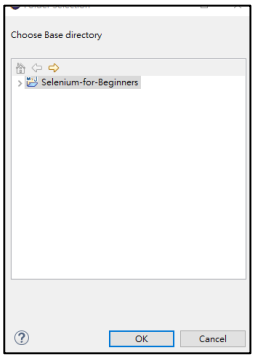
4. Use Eclipse environment to run mvn

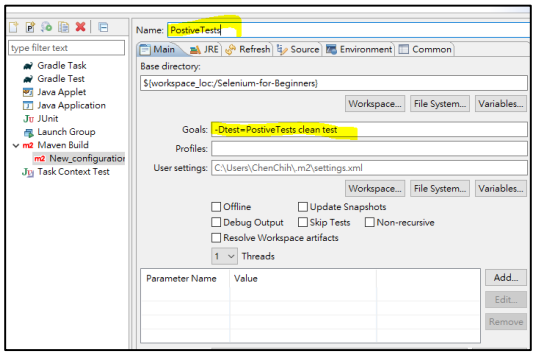
Set up MVN Environment configuration first

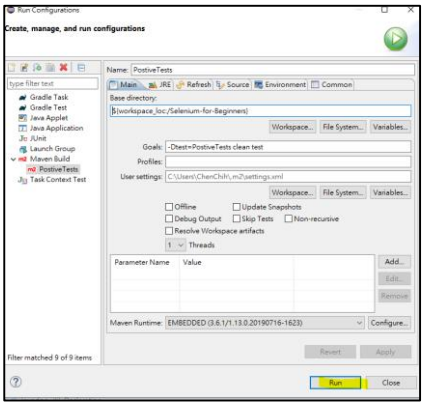


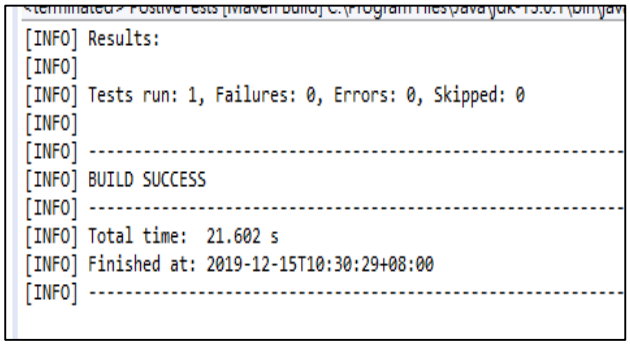



# Understand Locator

Locator tutorial:

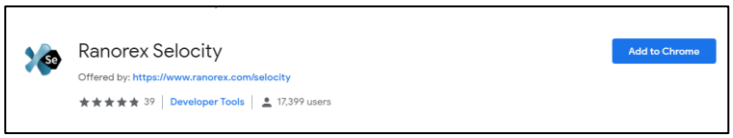
https://www.guru99.com/locators-in-selenium-ide.html

https://www.protechtraining.com/content/selenium_tutorial-locators

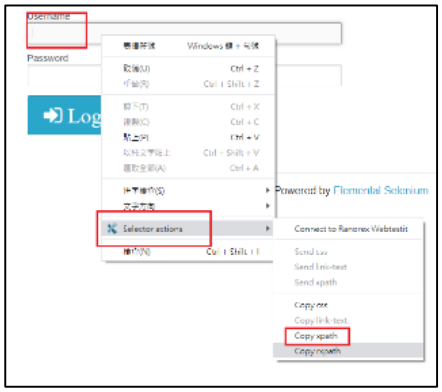
You can also use “Ranorex Selocity” plugin in chrome, it will automatic help we select the id.

Normal condition with chrome inspect


With Ranorex selocity plugin




It will automatic copy the id for you

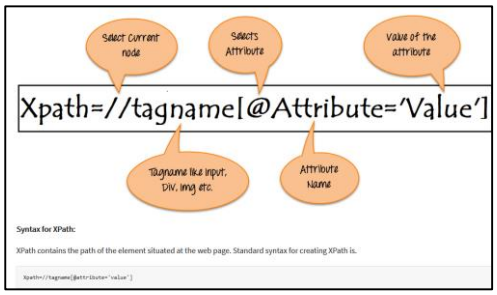


1. Xpath

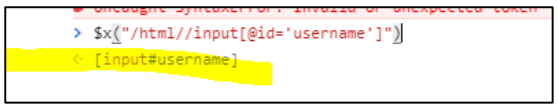
Xpath: //tag[@attrribue=’value’]

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

Css: input#username




use chrome console to check



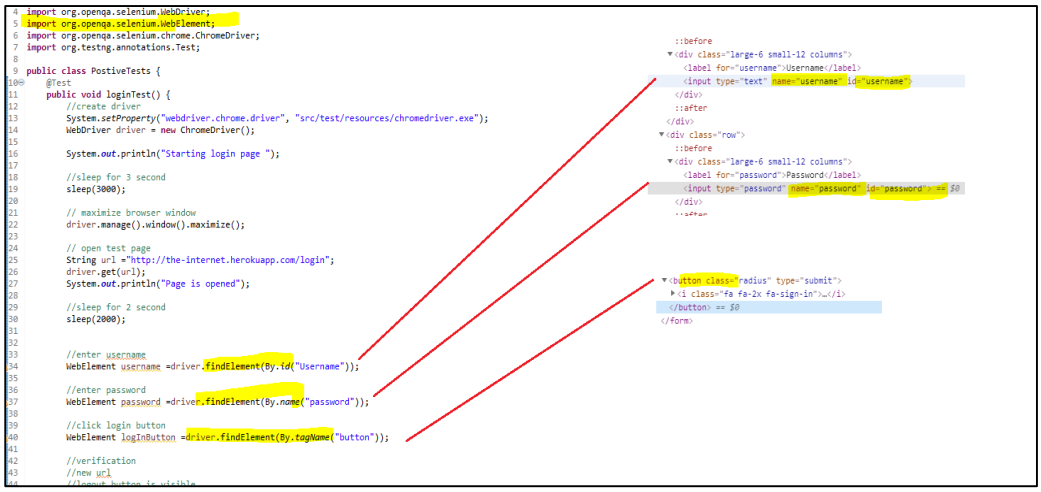

# Test Plan3 Create login user and password page

Description: Enter user and password

| **File used:** |
| --- |
| PostiveTests.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"));
```




2. Logout button (after login)

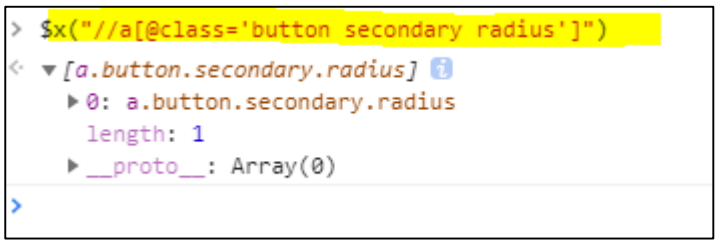
Find xpath value

Used console to find the xpath value

Go to inspect


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

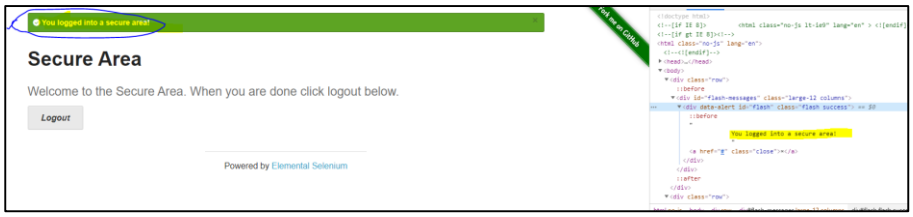
$x("//a[@class='button secondary radius']")




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

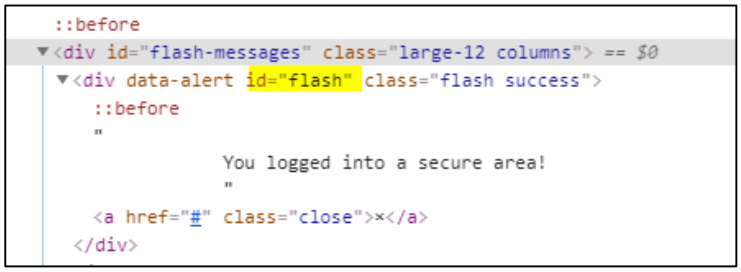
```
//logout button is visible
WebElement logOutButton =driver.findElement(By.xpath("//a[@class='button secondary radius']"));
```

3. successful login message

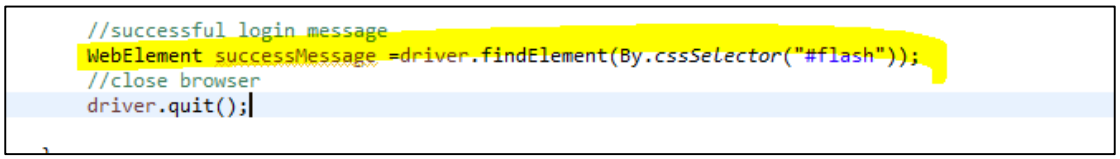



Use selector actions to copy css's div ("flash")


Or inspect




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



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

Verfication1- create assertion compare actual and expected

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

```
//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
```

```
[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] ------------------------------------------------------------------------
```

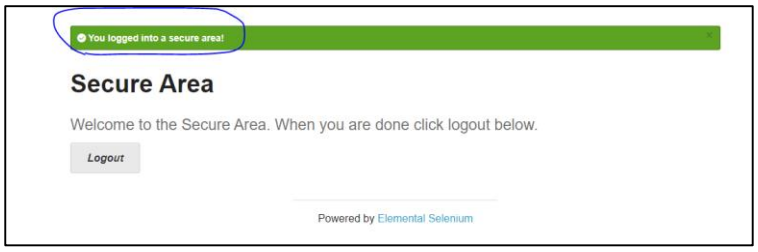
Verifcation2-verfiy logout button using Assert.True

```
//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)

```
        //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" );

        //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");
```

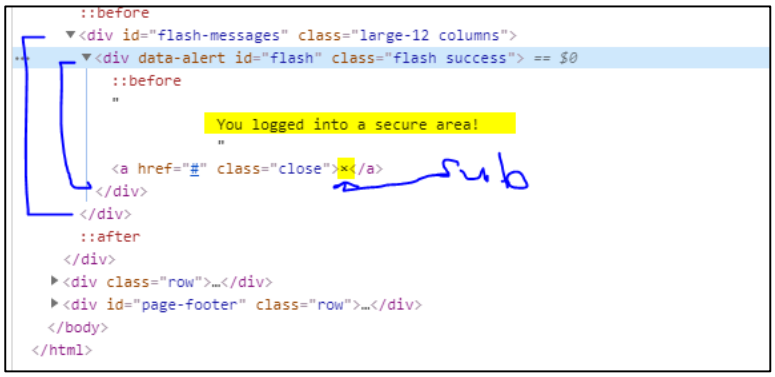
Expected Message is:




## 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.PostiveTests
[ERROR] com.herokuapp.theinternet.PostiveTests.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:



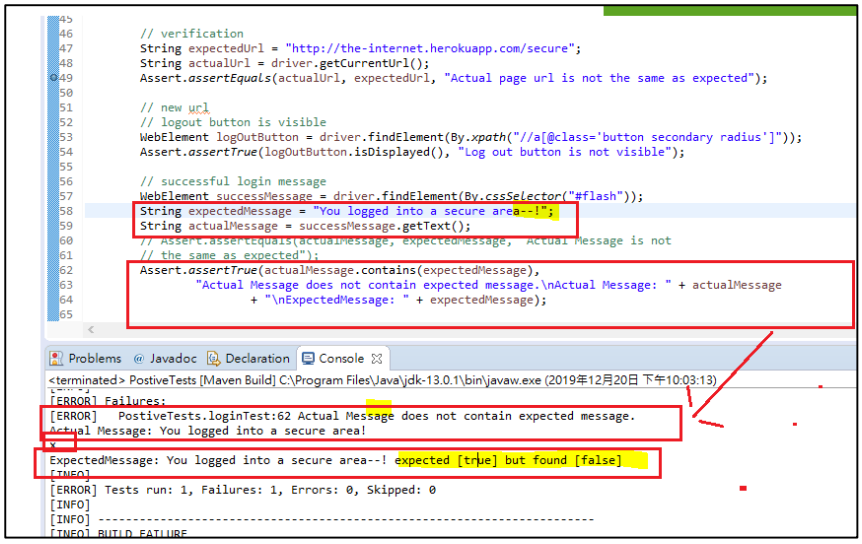

Look at the div element, it also contains a sub div element, X

Verification3 message (will be pass, correct mistake)

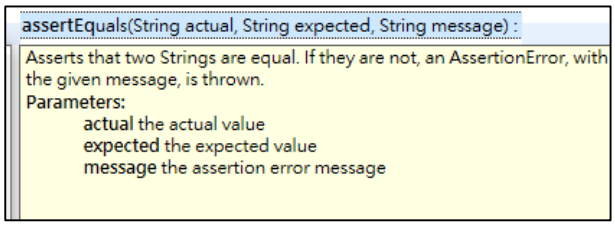
Using Assert.assertTrue yo solve this issue

```
        // 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
```

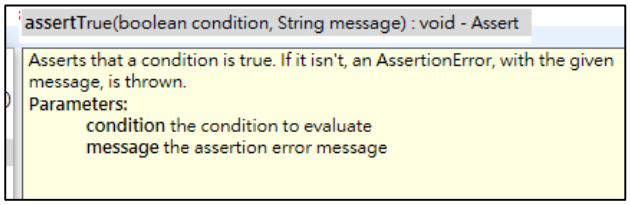
If fail will output like this, I try ro changed expected to --




6. Summary Assert




assertequals will print.




assertTrue will not print message, so we need to type in the message, but

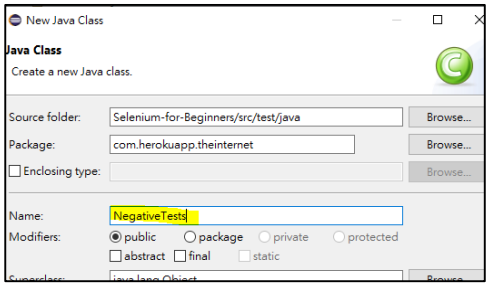
So since we want assert to be true, but it returns false.

# Test Plan4 Incorrect username test

Description: Enter incorrect user and password

| **File used:** |
| --- |
| PostiveTests.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

```
public class NegativeTests {
    @Test
    public void incorrectUsernameTest() {
                // create driver

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

                System.out.println("Starting login page ");

                // 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("SuperSecretPassword!");
                sleep(3000);
                // click login button
                WebElement logInButton = driver.findElement(By.tagName("button"));
                logInButton.click();
                sleep(5000);


    }
```

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");`

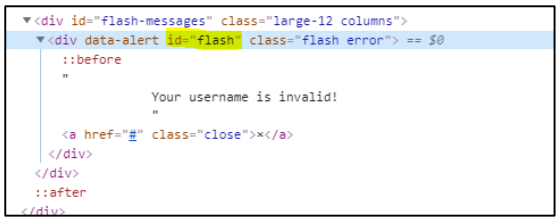
*and also changed*

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

4. Add verification

check error message if type in and inspect it


We will use id




```
            sleep(5000);
            //verfication
            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);
            //verfication
            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:** |
| --- |
| PostiveTests.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

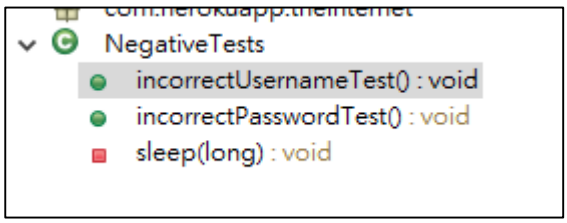
```
String expectedErrorMessage = "Your password is invalid!";
```

```java
public void incorrectPasswordTest() {
        System.out.println("Starting incorrecPassword 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);
        //verfication
        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

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

6. you can see the function

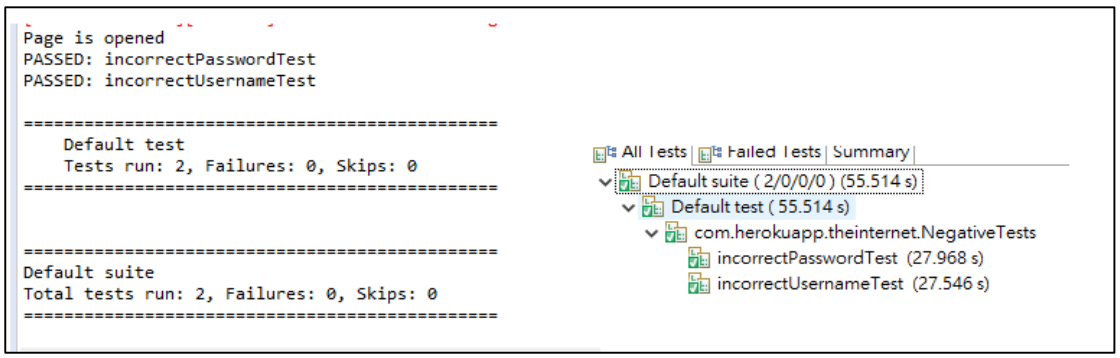



7. Run it

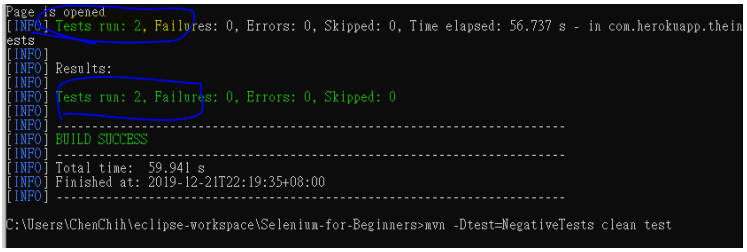
```
    Tests run: 1, Failures: 0, Skips: 0
================================================


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




9. Run Test under cmd




# 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 “

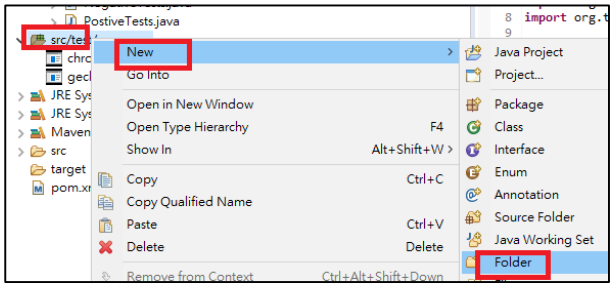
   -Go to above page and click “3 - testng.xml”

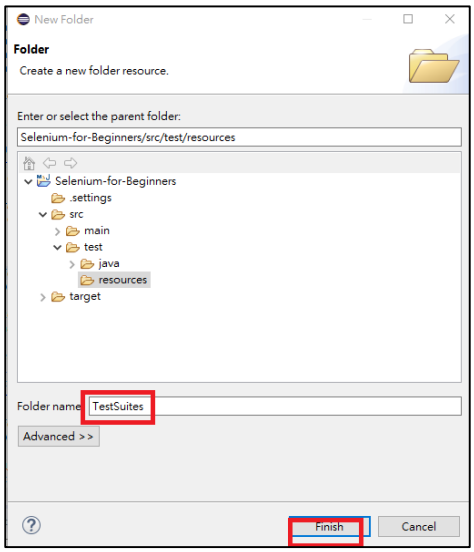
   -Copy the xml

e is an example testng.xml file:

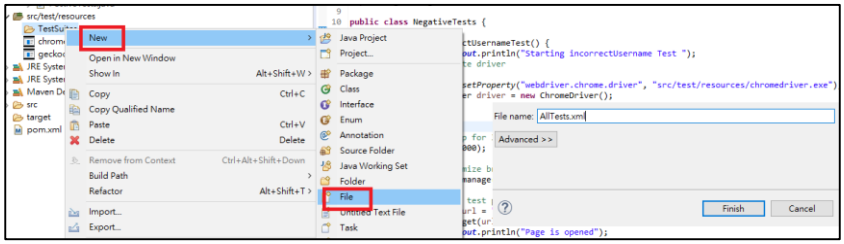
```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<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

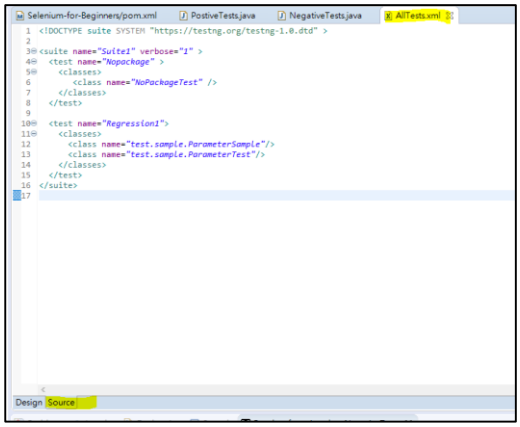



Add new File name AllTest.xml

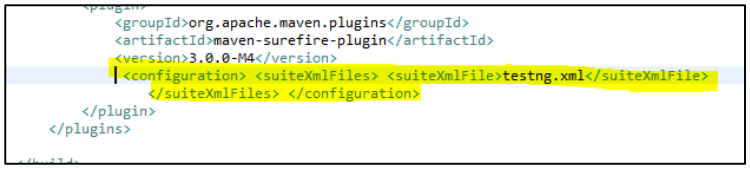


Go to source, and paste the url testin.xml content into source

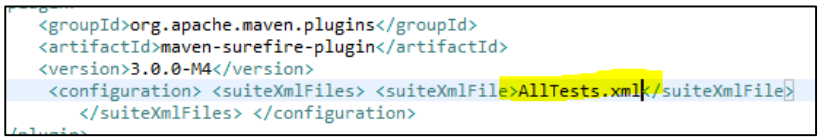



5. Edit Pom.xml file

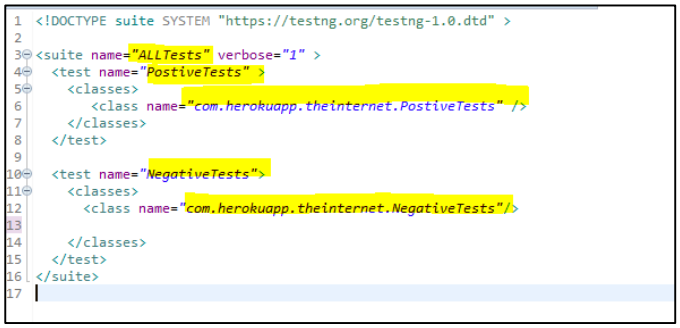
Uncomment configuration




Changed the testing.xml to AllTests.xml




6. Modify AllTest.xml




Since we have 2 class, so we add two class postiveTests, 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:**

PostiveTest.java: login success

NegativeTest.java: incorrect username and incorrect password

AllTest.xml: Test Suite

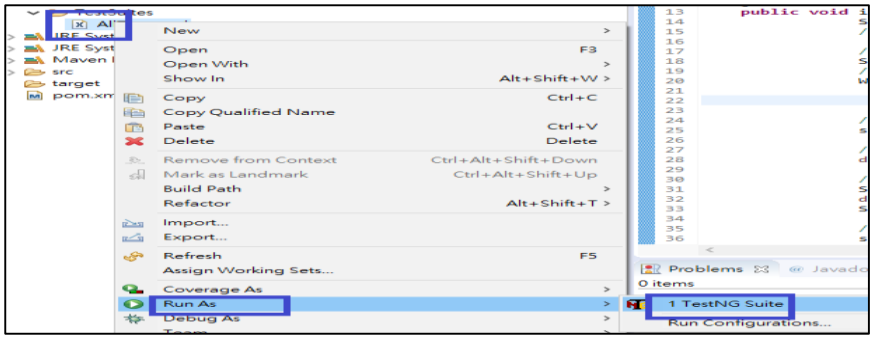
Folder name: case2-2-TestSuite

1. Running Test Suite

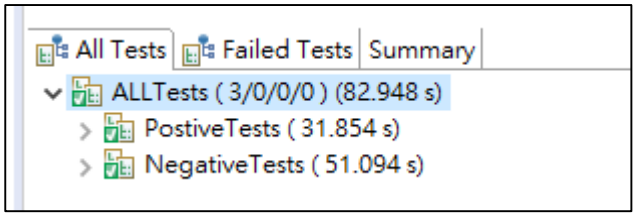
Changed only incorrectUsernameTest into Firefox driver, other just used chrome

```
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
```

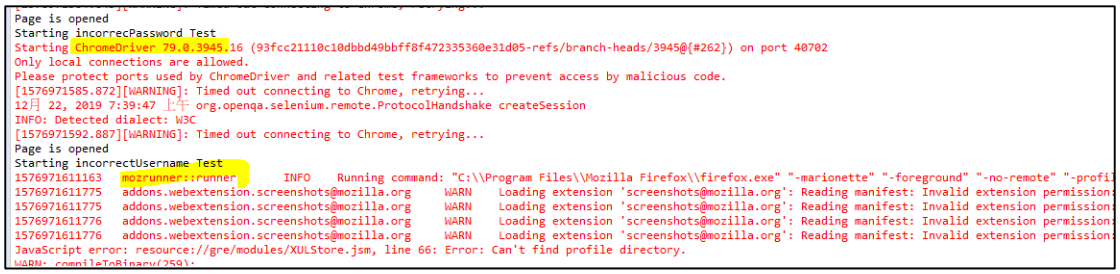
Run it right click Alltest.xml and run TestNG under Debug




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



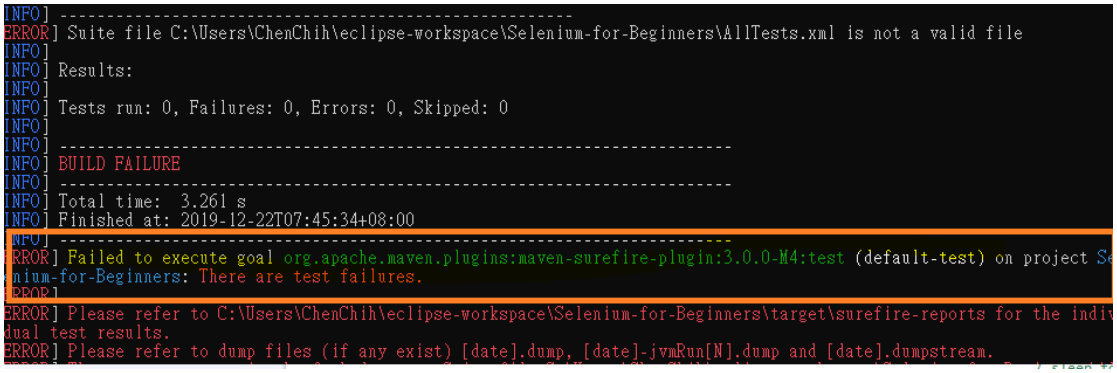

So you can see chrome and firefox is running below log



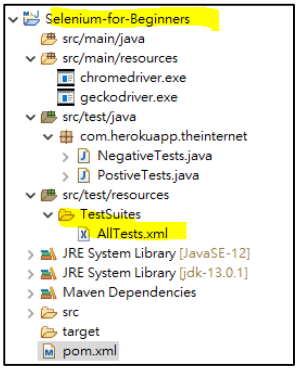

2. Run under cmd method

   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




   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:**

PostiveTest.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”

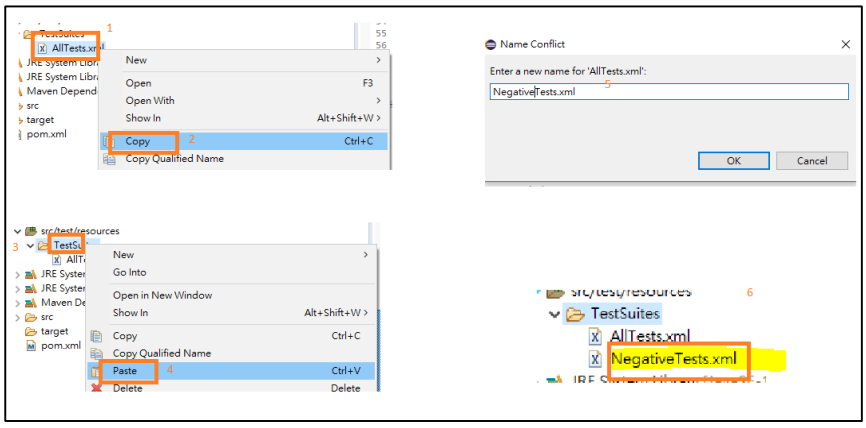
| @Test | Marks a class or a method as part of the test. |
|---|---|
| alwaysRun | If set to true, this test method will always be run even if it depends on a method that failed. |
| dataProvider | The name of the data provider for this test method. |
| dataProviderClass | The class where to look for 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, th |
| 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. |
| priority | 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 t |
| 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.<br>Note: this attribute is ignored if invocationCount is not specified |

4. Edit negativeTest.java

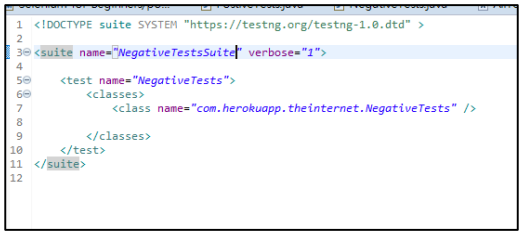
```
public class NegativeTests {
    @Test(priority =1)
    public void incorrectUsernameTest() {
                System.out.println("Starting incorrectUsername Test ");
                // create driver
```

```
 @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

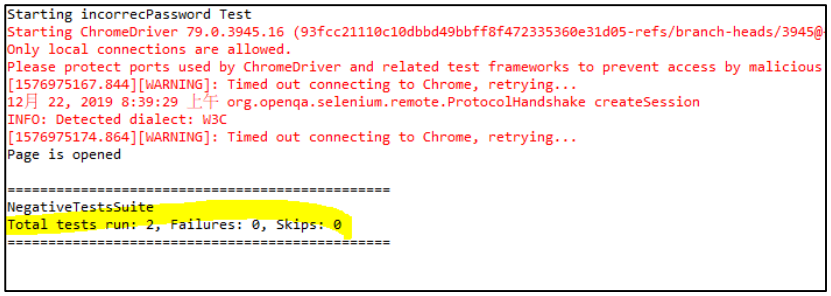


6. Modify NegativeTests.xml



```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<suite name="NegativeTestsSuite" verbose="1">

    <test name="NegativeTests">
        <classes>
            <class name="com.herokuapp.theinternet.NegativeTests" />

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

7. Run NegativeTest.xml



```
Starting incorrecPassword 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

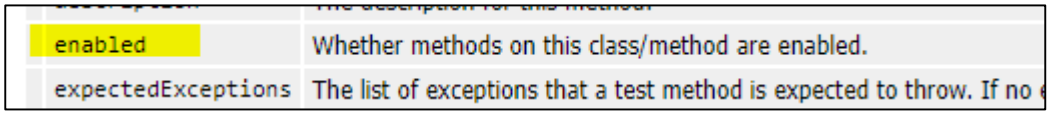
**File used:**

PostiveTest.java: login success

NegativeTests_2_enable.java: incorrect username and incorrect password

AllTest.xml: Test Suite

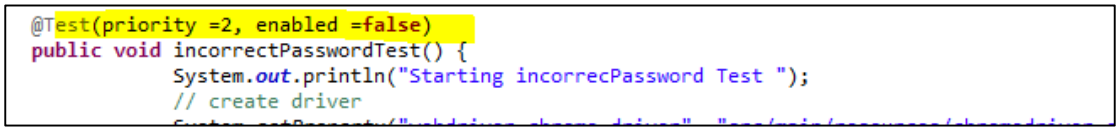
Folder name: case3-1-TestNG-Priority



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

1. Modify negativeTest.java

set @Test to disable like this enabled=false



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

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 ");
```

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”

```
<suite name="NegativeTestsSuite" verbose="1">

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

        <classes>
            <class name="com.herokuapp.theinternet.NegativeTests" />

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

3. Enable incorrectPassword to enable

   Remove `enabled =false`

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

   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 incorrecPassword Test ");
```

4. Run NegativeTests.xml and will only run incorrectUsernameTest

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

All Tests | Failed Tests | Summary

- NegativeTestsSuite ( 1/0/0/0 ) (22.906 s)
  - NegativeTests ( 22.906 s)
    - 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

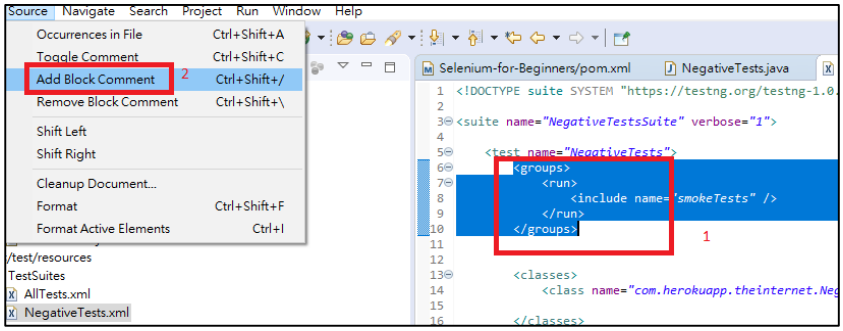
**File used:**

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



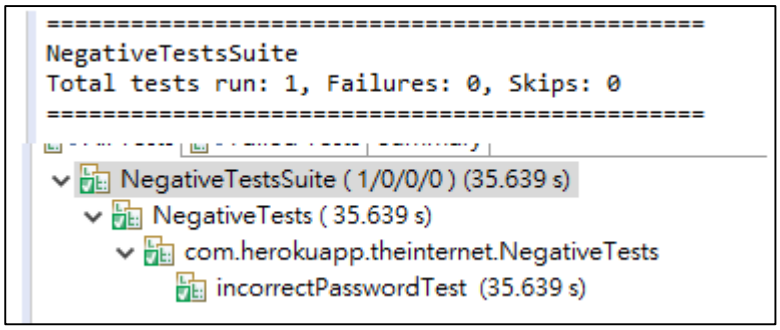

3. Change to include or exclude

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

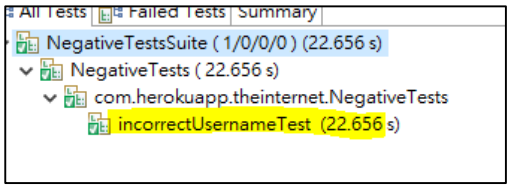
IncorrectPasswordTest will been execute




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 pssing parameter

| **File used:** |
| --- |
| NegativeTests.java: |
| NegativeTests.xml: Test Suite |
| Folder name: case4-1-passparmater |

Pass parameter to test method

1. Modify NegativeTest.xml

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

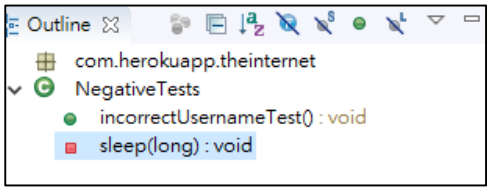
```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<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="tomsmith" />
        <parameter name="password" value="incorrectPassword!" />
        <parameter name="expectedMessage" value="Your password is invalid!" />
        <classes>
            <class name="com.herokuapp.theinternet.NegativeTests">

            </class>

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

2. Modify negativetest.java

comment incorrectPasswordTest this function, left with only one function

```
/*
 * @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.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); //verfication 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); //close browser
 * driver.quit(); }
 */
```

After we save, we will only see incorrectUsernameTest this function




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, Str
    System.out.println("Starting IncorrectUsernameTest");

//  Create driver
    System.setProperty("webdriver.gecko.driver", "src/main/resource
    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.");

//  enter username
    WebElement username = driver.findElement(By.id("username"));
    username.sendKeys("incorrectUsername");
```

String

webelement

6. Full Code

```
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 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();
                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
                WebElement logInButton = driver.findElement(By.tagName("button"));
                logInButton.click();
                sleep(5000);
                //verfication
                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
```

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:

```java
package com.herokuapp.theinternet;

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

public class NegativeTests {

    /*
     * @Test(priority =2, groups= { "negativeTests" }) 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();
     *
     * // 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);
     *
```

2. Paste to PostiveTest.java

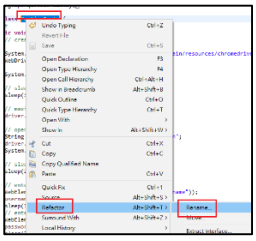
```java
    @Parameters({ "username", "password","expectedMessage"  })
    @Test(priority =1, groups= { "negativeTests", "smokeTests" })
    public void NegativeLoginTest(String username, String password, String expectedErrorMessage) {
            System.out.println("Starting negativeLogin Testwith "+username+"and"+password);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

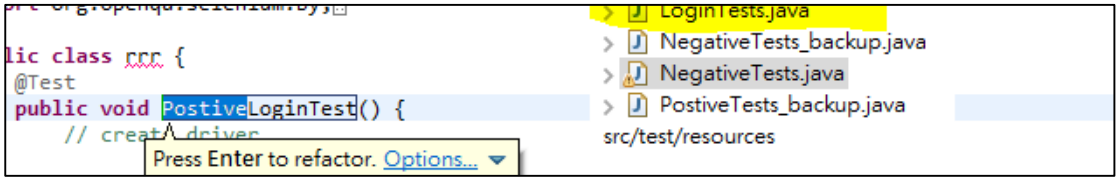
Rename `public void loginTest()` to `public void PostiveLoginTest()`

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

4. Rename file `postiveTests` to `LoginTest`



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



5. We can delete Negative Test.java if you want
6. Code LoginTest.java as below:

```
11
12 public class LoginTests {
13     @Test(priority =1, groups= { "postiveTests", "smokeTests" })
14     public void PostiveLoginTest() {
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 ");
```

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

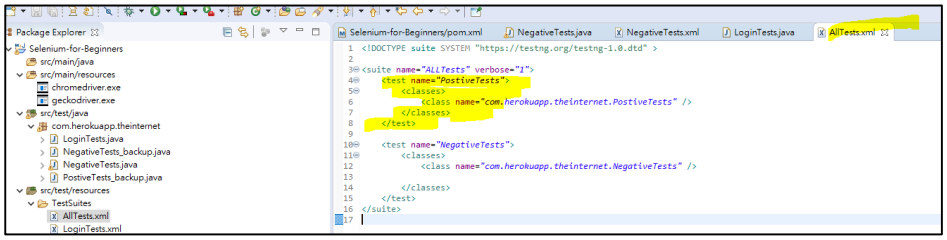
```
@Parameters({ "username", "password","expectedMessage"  })
@Test(priority =2, groups= { "negativeTests", "smokeTests" })
public void NegativeLoginTest(String username, String password, String expectedErrorMessage) {
            System.out.println("Starting negativeLogin Testwith "+username+"and"+password);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

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

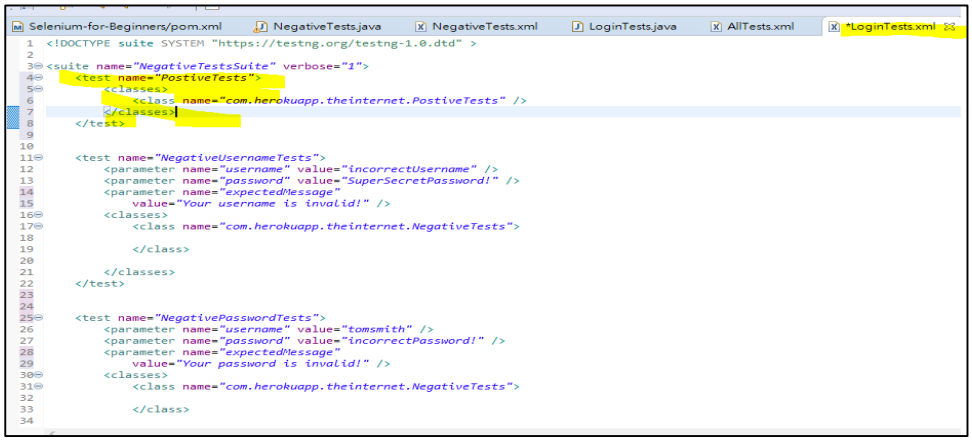
7. Create new xml suite LoginTest.xml (copy NegativeTests.xml)

Copy AllTests.xml's postiveTest.

AllTests.xml




LoginTest.xml




We can also delete AllTest.xml if you like

8. Add group to postiveTests

```
<suite name="NegativeTestsSuite" verbose="1">
    <test name="PostiveTests">
        <groups>
            <run>
                <include name="postiveTests"></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

LogininTest.XML

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<suite name="NegativeTestsSuite" verbose="1">
    <test name="PostiveTests">
        <groups>
            <run>
                <include name="postiveTests"></include>
            </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">
                <methods>
                    <include name="negativeLoginTest">
                    </include>
                </methods>
            </class>

        </classes>
    </test>
```

LogininTest.java

```
package com.herokuapp.theinternet;

import org.openqa.selenium.By;

public class LoginTests {
    @Test(priority =1, groups= { "postiveTests", "smokeTests" })
    public void PostiveLoginTest() {
        // create driver

        System.setProperty("webdriver.chrome.driver", "src/main/resources/c

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

    //System.setProperty("webdriver.chrome.driver", "src/main/resource
    System.setProperty("webdriver.gecko.driver", "src/main/resources/g
    //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";
```

# Part 3-2: Creating Test Suite-Driver 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

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

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

**File used:**

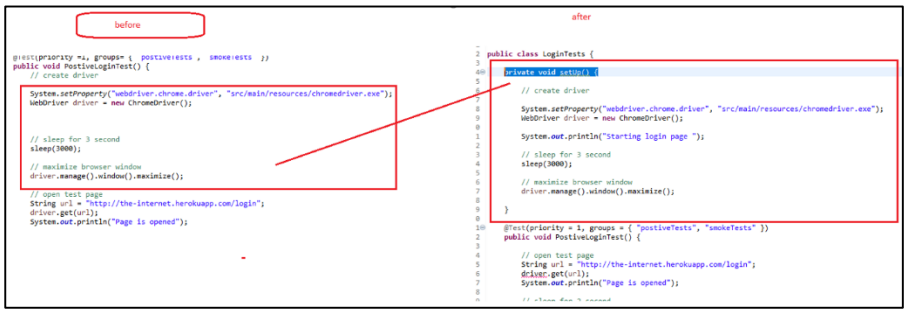
LoginTests.java: incorrect username and incorrect password

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



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();
    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
```

```
@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:

```
public class LoginTests {
    @BeforeMethod
    pr
```

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)

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

4. Check Problem

```
@Test(priority = 1, groups = { "postiveTests", "smokeTests" })
public void PostiveLoginTest() {

    // 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"));
    logInButton.click();
    sleep(5000);

    // 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");
```

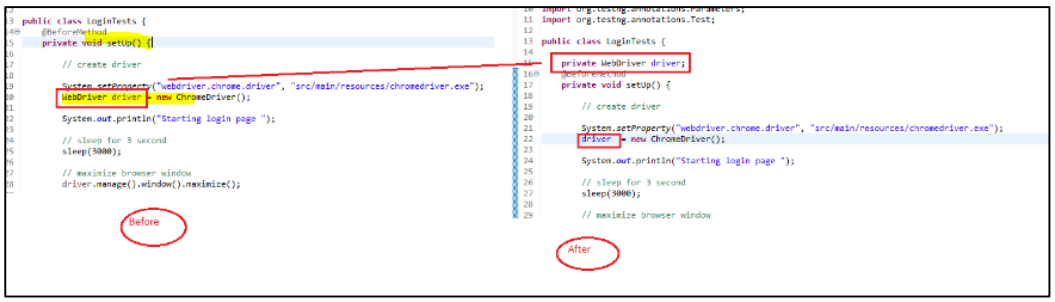
Problems | Javadoc | Declaration | Console | Results of running suite

14 errors, 8 warnings, 0 others

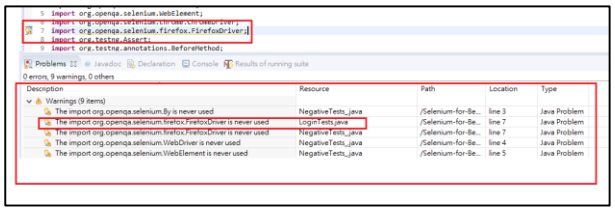
| Description | Resource | Path | Location | Type |
|---|---|---|---|---|
| driver cannot be resolved | LoginTests.java | /Selenium-for-Be... | line 37 | Java Problem |
| driver cannot be resolved | LoginTests.java | /Selenium-for-Be... | line 44 | Java Problem |
| driver cannot be resolved | LoginTests.java | /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:

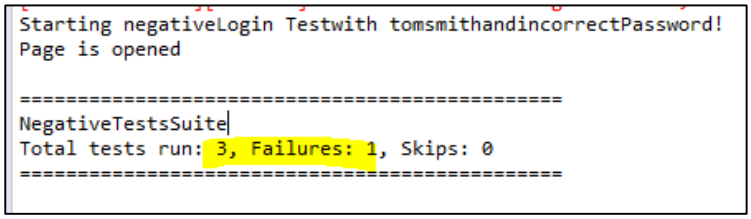
Move webdriver driver this variable and place under LoginTest and create a private variable as like this `private WebDriver driver;`



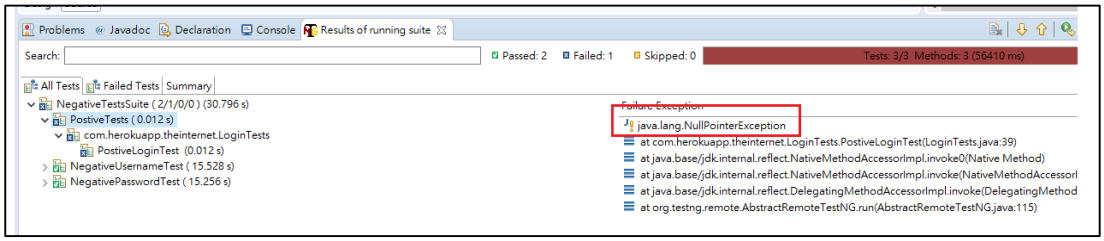

5. After saving it, look at problem, just left with warning, this is related to Firefox, just remove import firefox, then it will be fine



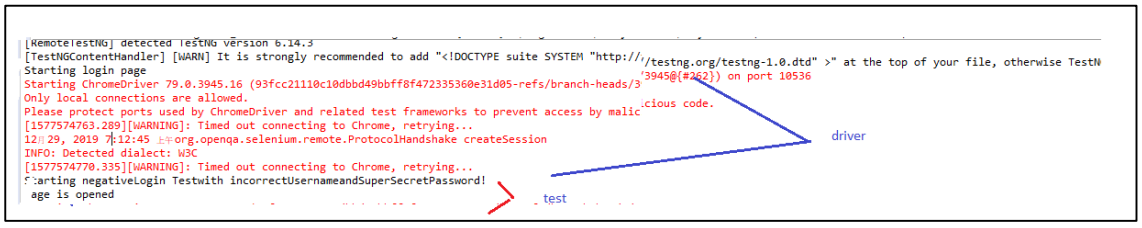
6. Run with Fail



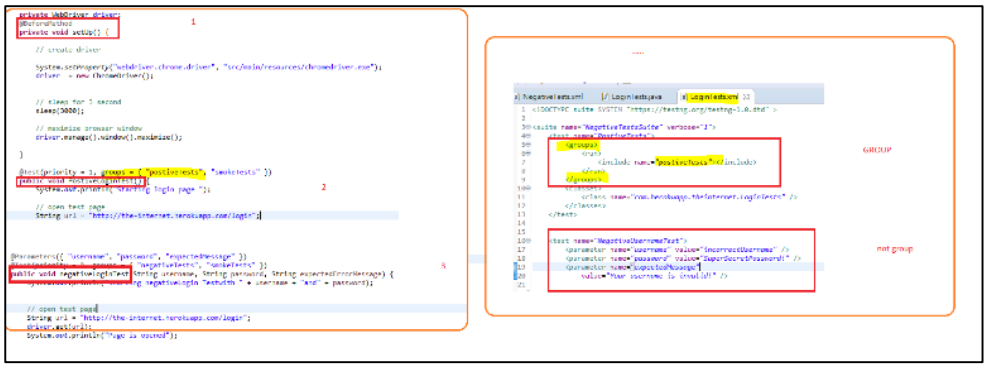

7. Fail and explanation




Driver will run first, then run method



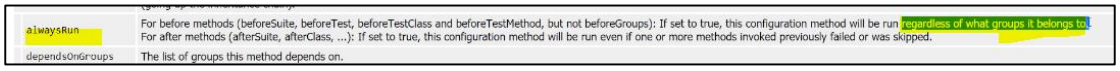

Driver will run first due to we add the Before Method



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

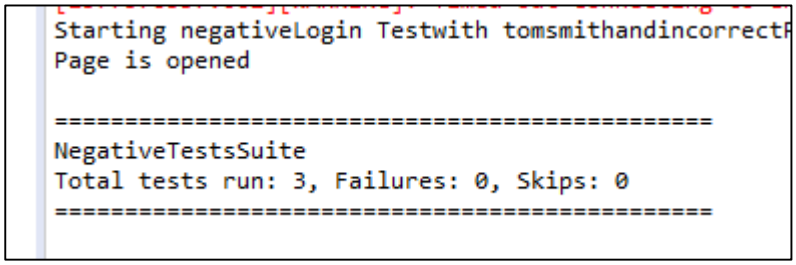
But the problem for PostiveTest is it's group, so it can't find it

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

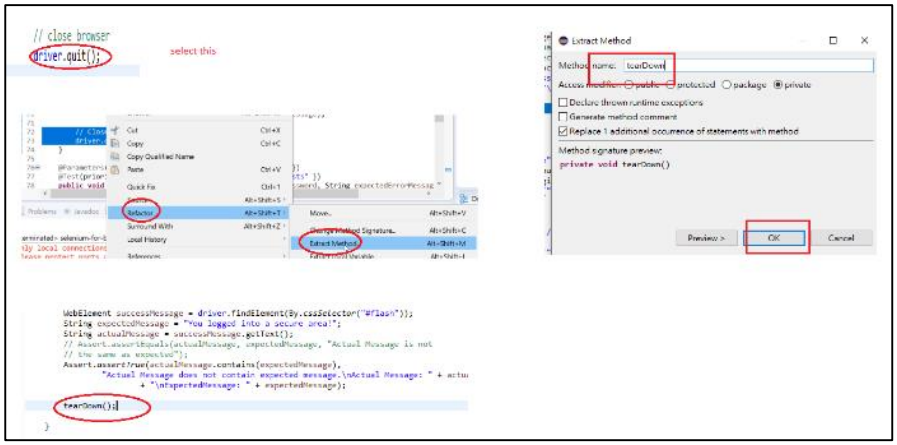



If adding this it won't care about is it in group or not

```
public class LoginTests {

    private WebDriver driver;
    @BeforeMethod(alwaysRun=true)
    private void setUp() {

        // create driver

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


        // sleep for 3 second
        sleep(3000);

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

    }

    @Test(priority = 1, groups = { "postiveTests", "smokeTests" })
    public void PostiveLoginTest() {
```



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 }
139
```

```
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:

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

public class LoginTests {

    private WebDriver driver;

    @Parameters({ "browser" })
    @BeforeMethod(alwaysRun = true)
    private void setUp(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 khow to start " + browser + ", starting chrome browser");
            System.setProperty("webdriver.chrome.driver", "src/main/resources/chromedriver.exe");
            driver = new ChromeDriver();
            break;

        }
```

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" />
```

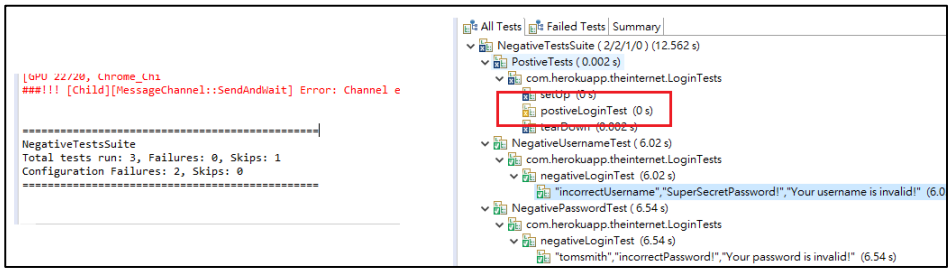
And changed to

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

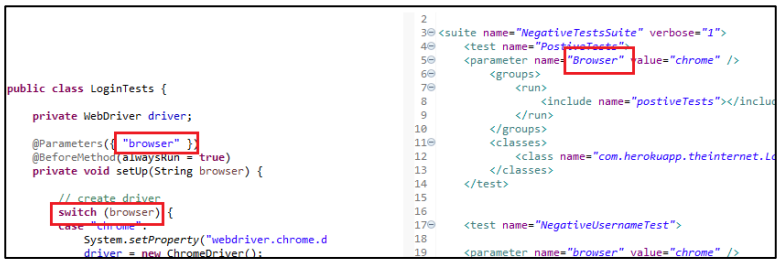
Do for both positiveTests and NegativeTests

```
<suite name="NegativeTestsSuite" verbose="1">
  <test name="PostiveTests">
  <parameter name="Browser" value="chrome" />
      <groups>
          <run>
              <include name="postiveTests"></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">
                  </include>
              </methods>
          </class>

      </classes>
  </test>


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

5. Test will fail



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




6. Add @optional to browser

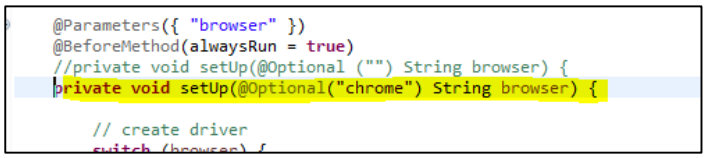
Add optional to browser in LoginTest.java, so it can skip capitalize.

Add

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

Or

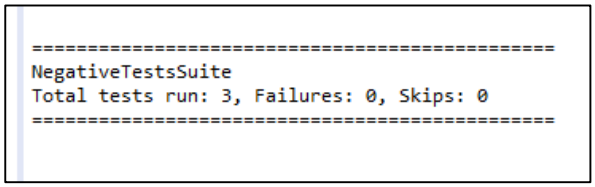
```
private void setUp(@Optional("") String browser)
```




7. Go to LoginTest.xml

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

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



# Part 4: Debugging

## Toggle breakpoint

1. Add toggle breakpoint and run with debug

Add debug.xml

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name="Debug" verbose="1">

    <test name="PositiveLoginTest">
        <parameter name="browser" value="chrome"></parameter>
        <classes>
            <class name="com.herokuapp.theinternet.LoginTests">
                <methods>
                    <include name="postiveLoginTest"></include>

                </methods>
            </class>
        </classes>
    </test>

</suite>
```

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

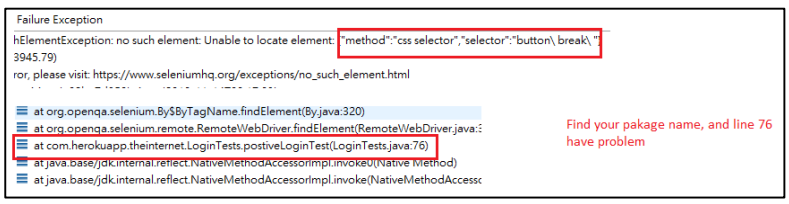
So add button break, since there is no this id, it will break

```
@Test(priority = 1, groups = { "postiveTests", "smokeTests" })
public void postiveLoginTest() {
    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.

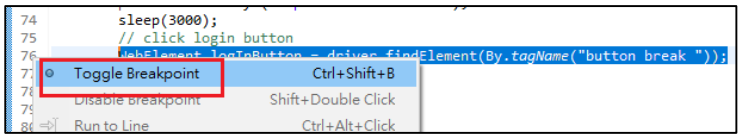
```
[1577801758.161][WARNING]: Timed out connecting to Chro
Starting login page
Page is opened

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

3. Debug message, how to see code have problem



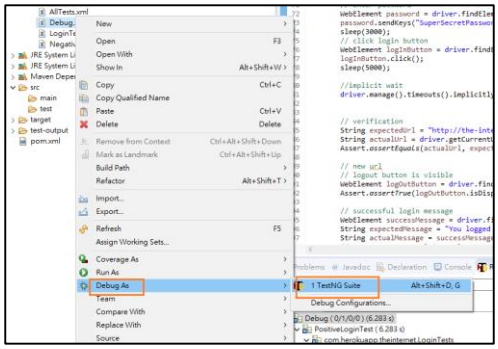
4. Sometime, it not easy to debug, we have to used debug toggle methods.

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

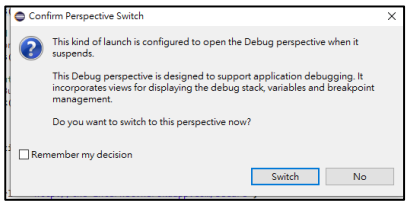



Or you can add by run>Toggle Breakpoint, or just double click it will occur a blue dot is ok.

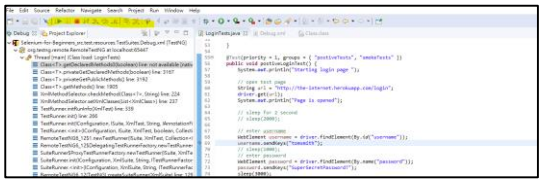
After set Toggle Break Point, then Run Debug



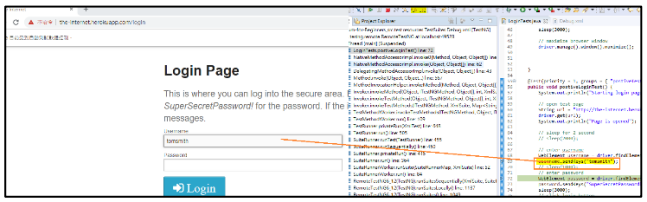

Press Switch

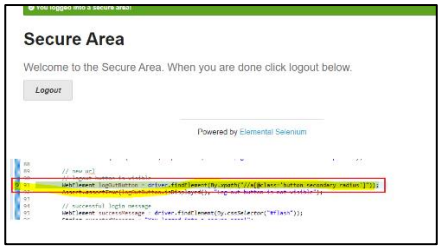



Debug will stop at toggle. This are debug option you can used

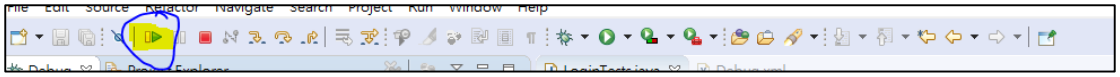


Press step over, will run one line by line






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_inplicit |

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 khow 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)

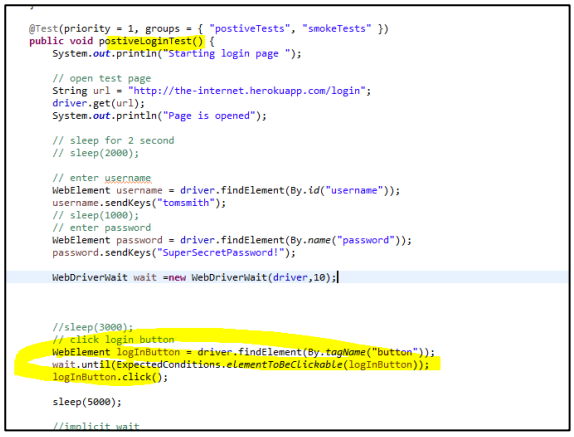
2. Using Explicit way Modify LoginTests.java

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 = { "postiveTests", "smokeTests" })
public void postiveLoginTest() {
    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 ElementNotVisibleException

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
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

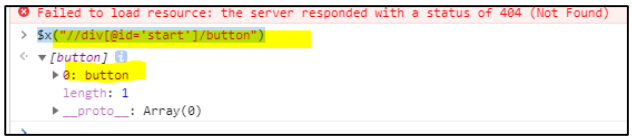
Add `public void notVisibleTest()`

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")




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();
```


5. compare using Assert

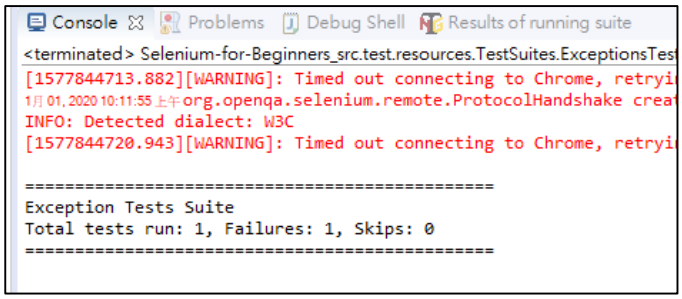
```
Assert.assertTrue(finishText.contains("Hello World!"));
```

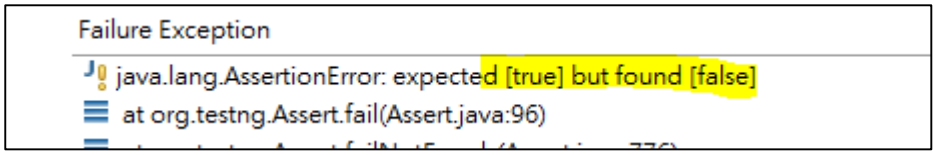
```
@Test
public void notVisibleTest() {
    //open the page http://the-internet.herokuapp.com/dynamic_loading/1
    driver.get("http://the-internet.herokuapp.com/dynamic_loading/1");
    //Find locator for start Button and click on it
    WebElement startButton=driver.findElement(By.xpath("//div[@id='start']/button"));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

7. Run test suite

It will Fail



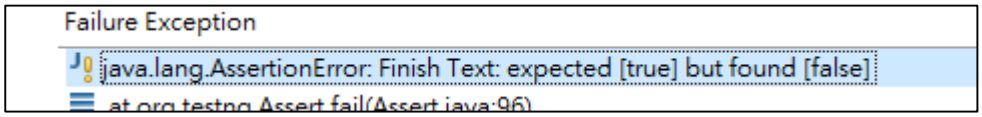



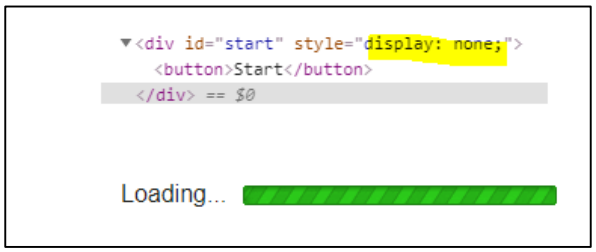

8. Debug it

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

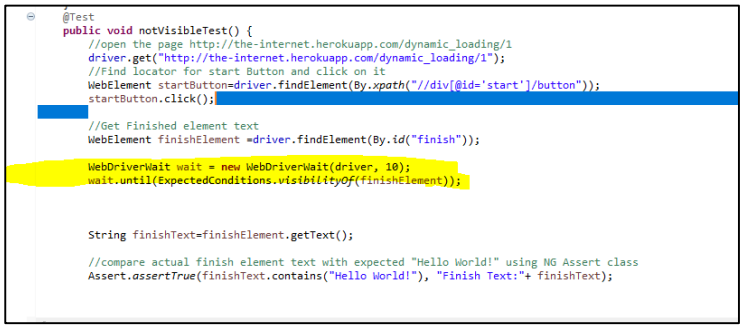



So we have to wait for a while, it will display. In this case we can't used implicit wait

Add this code, and wait 10 second

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



9. Run again

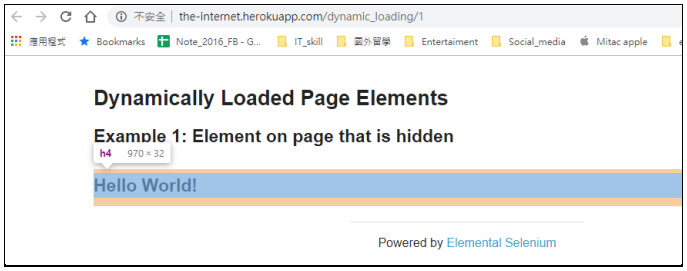
```
[1577852345.825][WARNING]: Timed out connecting to Chrome
1月 01, 2020 12:19:07 下午 org.openqa.selenium.remote.ProtocolHandsh
INFO: Detected dialect: W3C
[1577852352.869][WARNING]: Timed out connecting to Chrome

===============================================
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月 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
org.openqa.selenium.ElementNotInteractableException: element not interactable
  (Session info: chrome=79.0.3945.79)
```

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 Suite7 Timeout exception

Description:

**File used:**

ExceptionsTests.java:

ExceptionsTests.xml:

Folder name: exception-timout

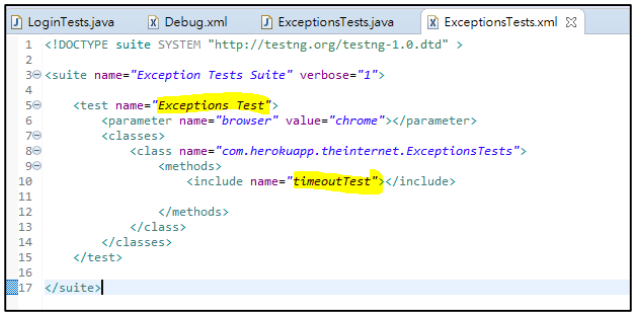
1. Changed `notVisibleTest()` to `timeoutTest()`

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();

}
```

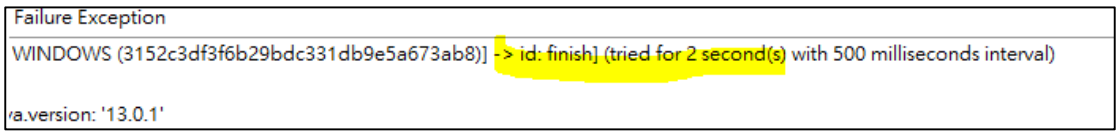
2. Modify Exception.xml file

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



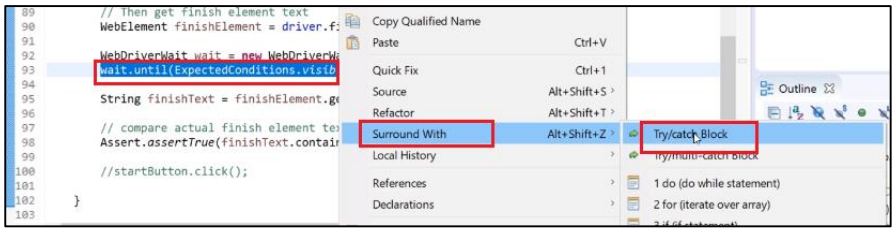
```
Failure Exception
WINDOWS (3152c3df3f6b29bdc331db9e5a673ab8)] -> id: finish] (tried for 2 second(s) with 500 milliseconds interval)

ra.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.

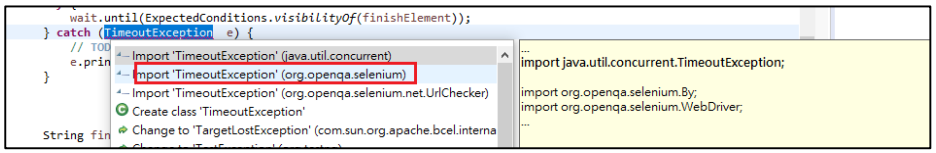



```
WebDriverWait wait = new WebDriverWait(driver, 2);
try {
    wait.until(ExpectedConditions.visibilityOf(finishElement));
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

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

Changed the general exception parameter

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



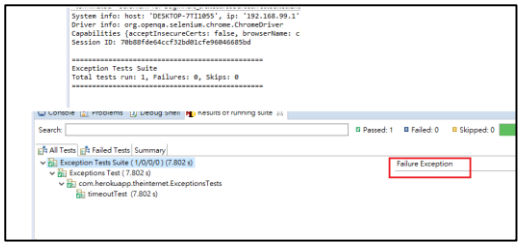

If exception is different than Timeout, Test will fail

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




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

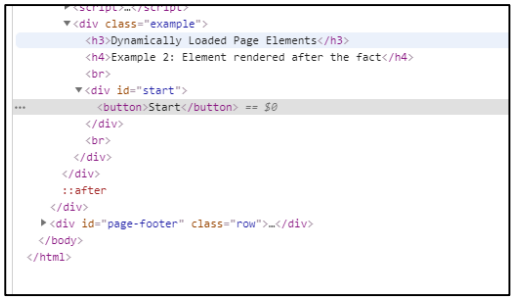
# Test Suite8 No suchElelementException

Description: when can't find element on page

| **File used:** |
| --- |
| ExceptionsTests.java: |
| ExceptionsTests.xml: |
| Folder name: exceptioon_noEleemnt |

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




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

```java
    @Test
    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);

    }
```

Go to Exception.xml and add the name to it

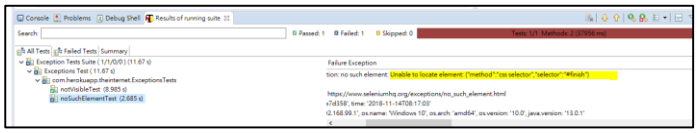
Since only three case, we will not run timeout, so we type in exclude timeoutTest

```xml
<!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>
                    <exclude name="timeoutTest"></exclude>

                </methods>
            </class>
        </classes>
    </test>

</suite>
```

2. changed the priority as below

```java
@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

```java
@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 m) {
```

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 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"></parameter>
        <classes>
            <class name="com.herokuapp.theinternet.ExceptionsTests">
                <methods>
                    <include name="noSuchElementTest"></include>

                </methods>
            </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("finish"), "Hello World!")),
"Could't verify expected text 'Hello World!'");
```

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

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 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:

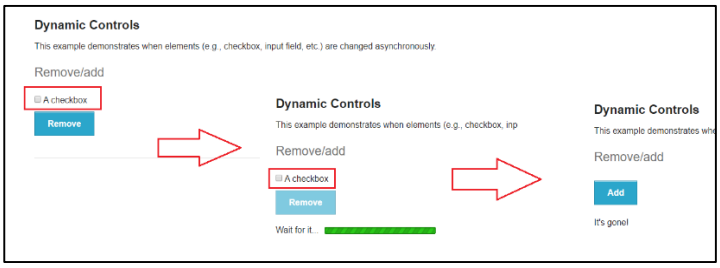
# Test Suite9: Tricky Exception:

## StaleElementReferenceException

Description: The element have been remove

| **File used:** |
| --- |
| ExceptionsTests.java: |
| ExceptionsTests.xml: |
| Folder name: slame |

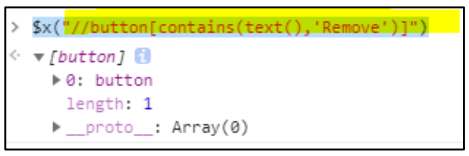
Test Link: http://the-internet.herokuapp.com/dynamic_controls



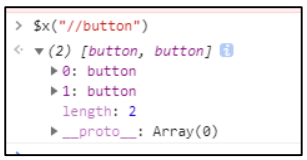

1. Create a method public void staleElementTest()
2. Find the correct locator to used

Go to chrome and inspect console page and type in

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



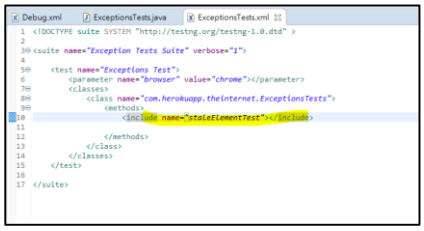

Two button



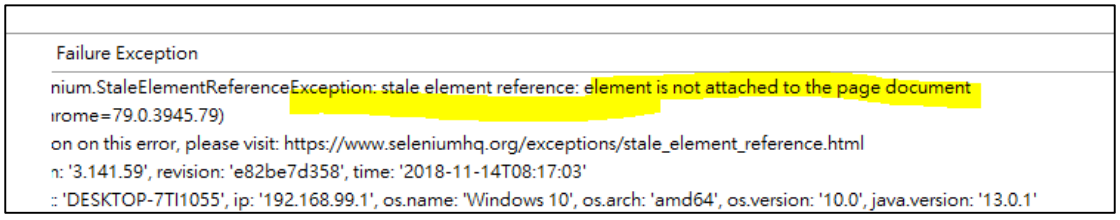
3. Add the code in staleElementTest

```
@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



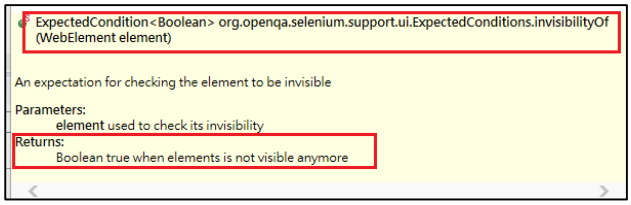

5. Run it and fail




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

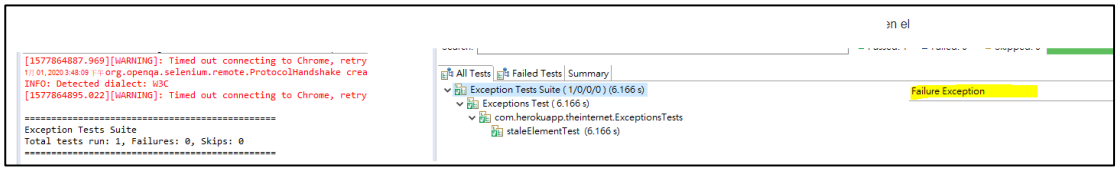
Comment out this code first

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

Add below code:

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

Run it will pass

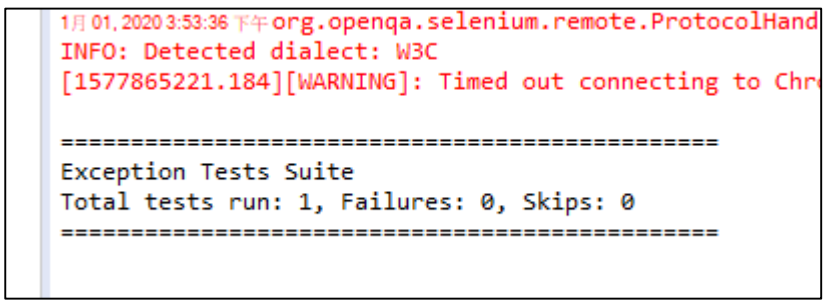



Note: not all case will pass

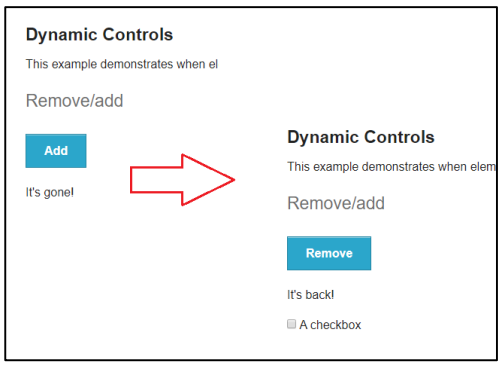
6. Another method

Comment the code again

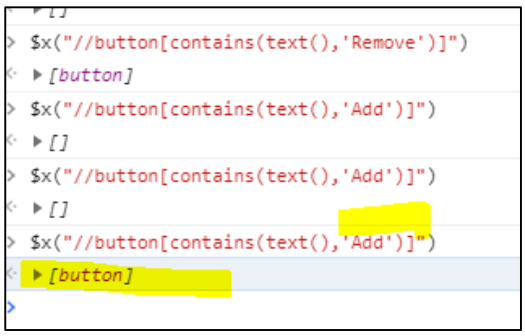
```
Assert.assertTrue(wait.until(ExpectedConditions.stalenessOf(checkbox)),
"CheckBox is still visible, but shouldn't be");
```



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




Inspect checkbox




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 visble,
but it shiould be  ");
```

```java
@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 shouln'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(), "Checkbox is not visble, but it shiould be  ");

}

private void sleep(long m) {
    try {
        Thread.sleep(m);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
```

Will fail, most common people will face on this problem

```
Failure Exception
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:
```

Because checkbox is remove 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 visble, but it shiould be  ");
```

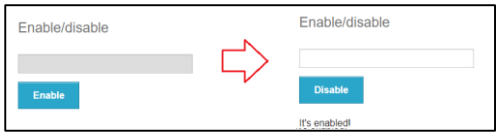
Add this

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

```java
@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 shouln'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(), "Checkbox is not visble, 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
===============================================
```

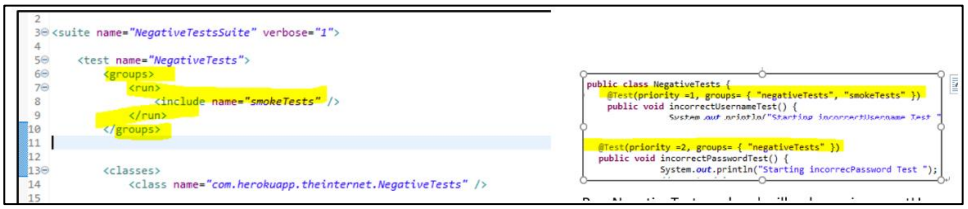
5. This is same as previous link, just




# Part 5: Summary

## Test Suite-Summarize All

1. Using group:



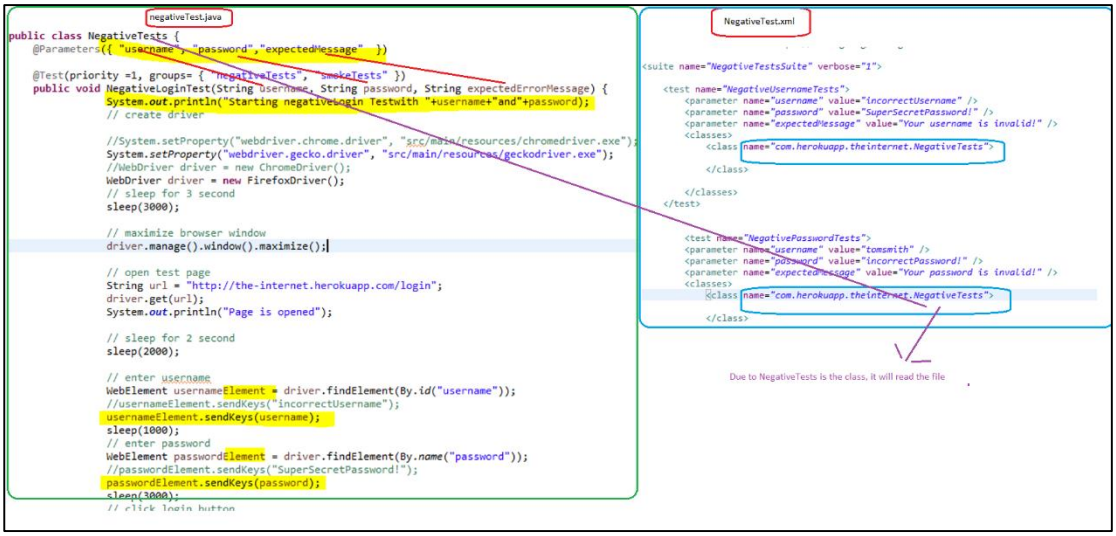

Will run incorrectUsernameTest, because it have smokeTest as group

2. using Include:

Run incorrectUsernameTest, when <u>exclude</u> incorrectPasswordTest


Run incorrectPasswordTest, when include incorrectPasswordTest

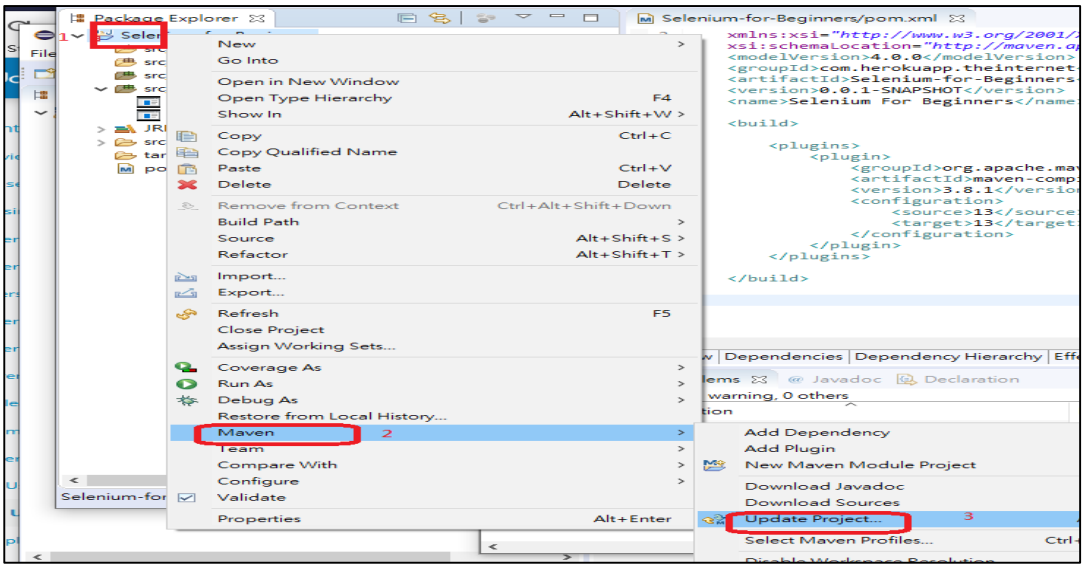
3. using Parameter




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


# Part 6: Reference

## 1.Update Marven Project


Package Explorer
Selenium-for-Beginners/pom.xml
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Assign Working Sets...
Coverage As
Run As
Debug As
Restore from Local History...
Maven
Team
Compare With
Configure
Validate
Properties Alt+Enter
Add Dependency
Add Plugin
New Maven Module Project
Download Javadoc
Download Sources
Update Project...
Select Maven Profiles...
Dependencies
Dependency Hierarchy
Javadoc
Declaration
1
2
3