TestNG

Session – 26

What is TestNG :-

**TestNG** is an automation testing framework in which NG stands for “**Next Generation”** . TestNG is inspired by Junit which uses the annotations (@).

Feature of TestNG we can use in selenium :-

* WebDriver has no native mechanism for generating reports. TestNG can generate the report in a proper & readable format.
* Multiple test cases can be grouped more easily.
* The same test case can be executed multiple times without loops
* Using testNG, you can execute multiple test cases on multiple browser.
* The TestNG framework can be easily integrated with tools like TestNG Maven, Jenkins, etc.
* Annotations used in the testing are very easy to understand example :-

@BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest

Points to remember for writing TestNG testcase:-

1. TestNG does not require you to have a main() method.
2. Methods need not be static.
3. We used the @Test annotation, @Test is used to tell that the method under it is a test case.
4. We needed to import the package org.testng.annotations.\*;
5. We used the Assert class.

“Assert class is used to conduct verification operations in TestNG. Assertions in TestNG are a way to verify that the expected result matches the actual result”

To use it, we need to import the org.testng.Assert package

**package** TestNGdemo;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** io.github.bonigarcia.wdm.WebDriverManager;

**import** org.testng.Assert;

**import** org.testng.annotations.\*;

**public** **class** TestNGdemo1 {

@Test

**public** **void** verifyPageTitle()

{

//Launching chrome browser

WebDriverManager.*chromedriver*().setup();

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

//open url

driver.get("https://www.google.com/");

String actualTitle = driver.getTitle();

String expectedTitle = "Google";

// dono ko verify krna hai uske liye Assert class ka use krenge

Assert.*assertEquals*(actualTitle, expectedTitle);

driver.quit();

}

}

Session – 27

What is TestNG XML Flie :-

In TestNG framework, we need to create Testng xml file to run and handle multiple test classes. TestNG.xml file is an XML file which contains all the test configuration and this XML file can be used to run and organize our test.

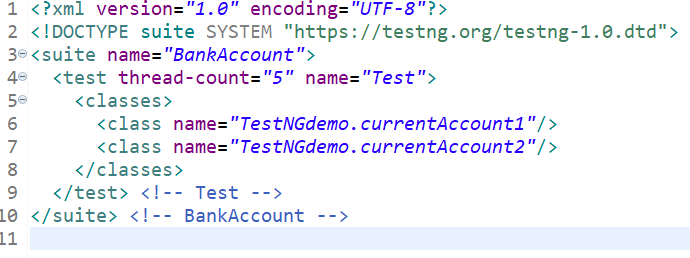
**In testing.xml file**, we configure our test run, set test dependency, include or exclude any test, method, class or package and set priority etc.

Testng.xml file is also is used for TestNG parameters. TestNG parameters are the arguments that we pass to the test methods

Steps :-

* Testng.xml create krne ke liye project pr right click krte hai or TestNG m jake convert to TestNG krte hai
* Eska use ek se jayada class ke method ko ek sath execute krne ke liye krte hai esme class tag m un class ka path dete hai unko execute krna ho
* Run ke liye testng.xml file m he right click krte hai or run as TestNG suite pe krte hai

Example :- current account wala



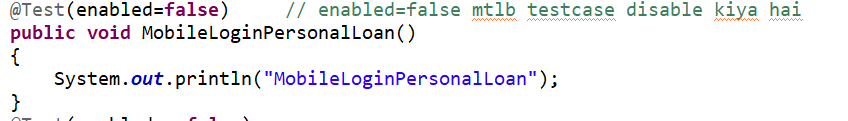
Session – 28

Enable and disable test cases :-

In TestNG, **test case can be enabled / disabled in two ways**

* **You can disable the test case in a @Test annotation**

TestNG @Test enable parameter



* **You can disable the test case in the XML file**

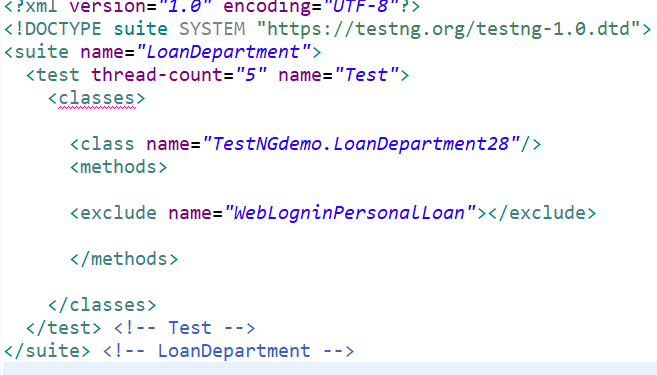
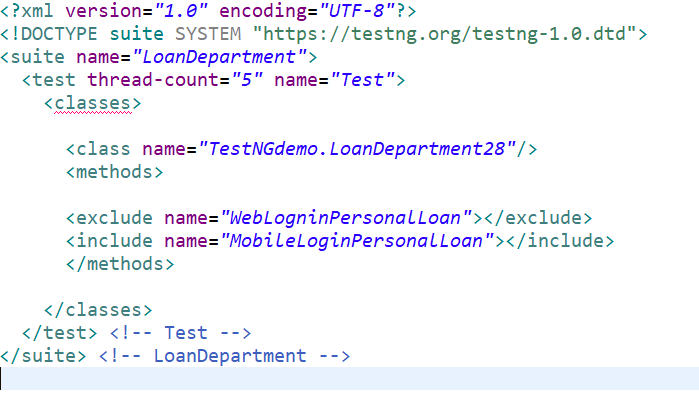
<methods>

<exclude name = “MethodName”/>

<include name = “MethodName”/>

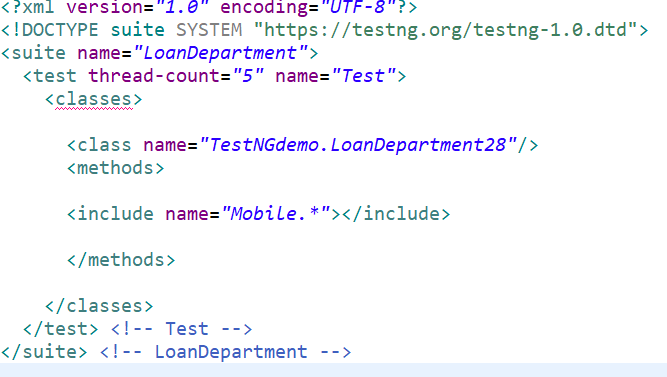
</methods>

* Exclude tag se disable hoga -> include tag se whi run hoga jisko include kiya ho

Using Regular expression –

Example – we method run krne hai jo “Mobile” se start ho



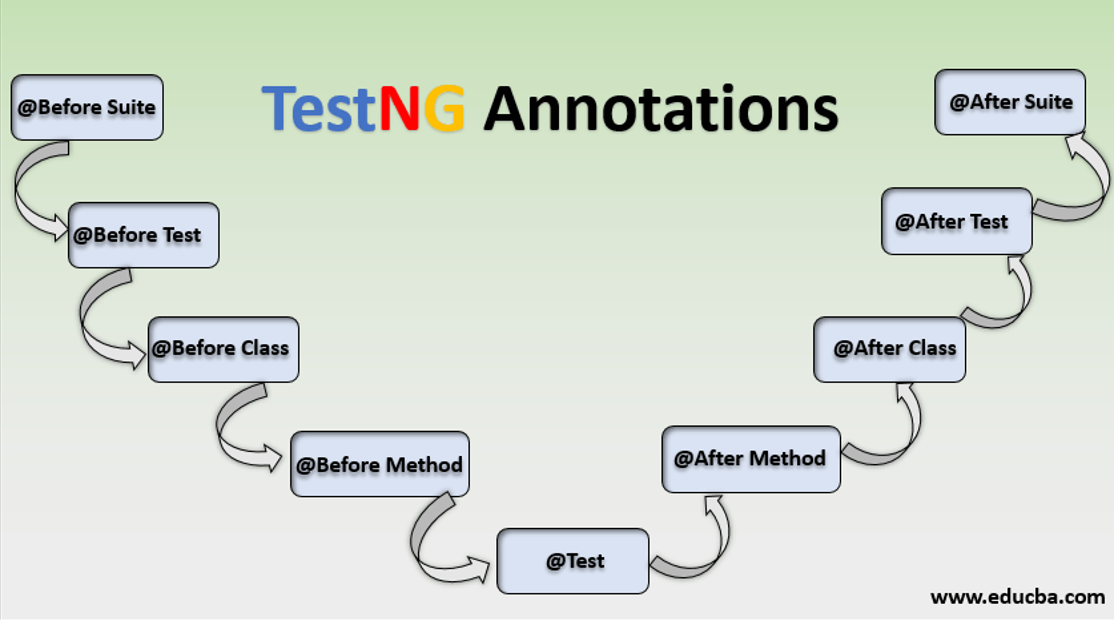
Session – 29

What are TestNG Annotation :-

An annotation is a tag/piece of code that provides additional information about the method. It is represented by ‘@’ prefix. It is used to control the execution of test cases

**Following are some of the benefits of using annotations :-**

* TestNG identifies the methods it is interested in, by looking up annotations. Hence, method names are not restricted to any pattern or format
* We can pass additional parameters to annotations
* Annotations are strongly typed, so the compiler will flag any mistakes right way



**@BeforeMethod :-** agr ek class m 4 method hai to unke run hone se pahle ye run hoga means- sbse pahle ye execute hoga fir test1 execute hoga fir dubara se before method execute hoga fir test2 execute hoga……

**@AfterMethod :-** ek class m jitne method hai sbke run hone ke baad ek ek baar ye execute hoga

Example – ek class m 2 method hai to pahle test1 execute hoga fir After mothod execute hoga uske baad test2 execute hoga fir AfterMethod execute hoga.

**@BeforeClass :-** class m sb method execute hone se pahle ek baar pahle BeforeClass method execute hoga uske baad sb honge

Example – agr 4 test method hai or ek Before class method to sbse pahle ek baar BeforeBlass method run hoga uske baad test1,test2,test3,test4 sb execute honge.

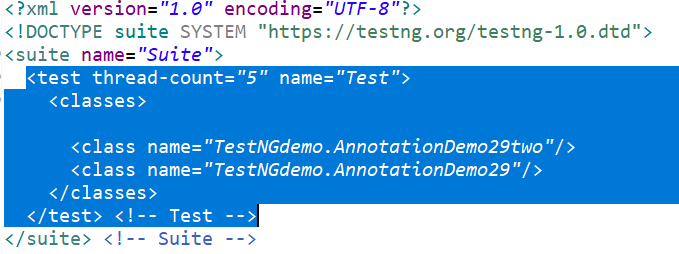
**@AfterClass :-** class m sb method execute hone ke baad AfterClass method execute hoga

Example – agr 4 test method hai or ek After class method to sbse pahle test1,test2,test3,test4 sb execute honge uske baad ek baar AfterClass method execute hoga.

|  |  |
| --- | --- |
| **package** TestNGdemo;  **import** org.testng.annotations.AfterClass;  **import** org.testng.annotations.AfterMethod;  **import** org.testng.annotations.BeforeClass;  **import** org.testng.annotations.BeforeMethod;  **import** org.testng.annotations.Test;  **public** **class** AnnotationDemo28 {      @BeforeClass  **public** **void** beforeClass()  {  System.***out***.println("BeforeClass");  }    @AfterClass  **public** **void** afterClass()  {  System.***out***.println("AfterClass");  }      @BeforeMethod  **public** **void** beforeMethod()  {  System.***out***.println("BeforeMethod....");  }    @AfterMethod  **public** **void** afterMethod()  {  System.***out***.println("AfterMethod....");  }    @Test  **public** **void** test1()  {  System.***out***.println("this is test1.....");  }    @Test  **public** **void** test2()  {  System.***out***.println("this is test2.....");  }    @Test  **public** **void** test3()  {  System.***out***.println("this is test3.....");  }    @Test  **public** **void** test4()  {  System.***out***.println("this is test4.....");  } | **Output** |

**@BeforeTest :-** dono class ke execution se pahle agr koi method execute krna ho to @BeforeTest Annotation ka use krenge.

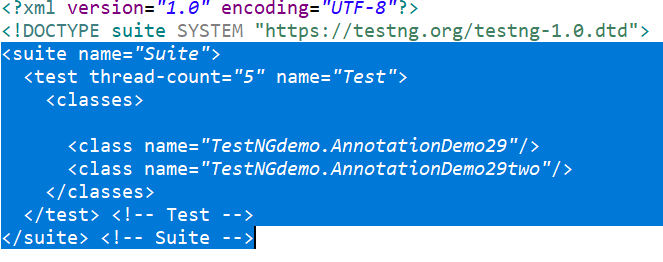
**@AfterTest :-**  dono class ke execution ke baad koi method execute krna ho to @AfterTest Annotation ka use krenge.



|  |  |
| --- | --- |
| package TestNGdemo;  import org.testng.annotations.AfterClass;  import org.testng.annotations.AfterMethod;  import org.testng.annotations.AfterTest;  import org.testng.annotations.BeforeClass;  import org.testng.annotations.BeforeMethod;  import org.testng.annotations.BeforeTest;  import org.testng.annotations.Test;  public class AnnotationDemo29 {    @BeforeTest  public void beforeTest()  {  System.out.println("Before Test method......");  }    @AfterTest  public void afterTest()  {  System.out.println("After Test method......");  }      @BeforeClass  public void beforeClass()  {  System.out.println("BeforeClass");  }    @AfterClass  public void afterClass()  {  System.out.println("AfterClass");  }      @BeforeMethod  public void beforeMethod()  {  System.out.println("BeforeMethod....");  }    @AfterMethod  public void afterMethod()  {  System.out.println("AfterMethod....");  }    @Test  public void test1()  {  System.out.println("this is test1.....");  }    @Test  public void test2()  {  System.out.println("this is test2.....");  }    @Test  public void test3()  {  System.out.println("this is test3.....");  }    @Test  public void test4()  {  System.out.println("this is test4.....");  }    } | **Output** |

**@BeforeSuite** :- ye suite se pahle execute hoga

**@AfterSuite :-** ye suite ke baad execute hoga



|  |  |
| --- | --- |
| package TestNGdemo;  import org.testng.annotations.AfterClass;  import org.testng.annotations.AfterMethod;  import org.testng.annotations.AfterSuite;  import org.testng.annotations.AfterTest;  import org.testng.annotations.BeforeClass;  import org.testng.annotations.BeforeMethod;  import org.testng.annotations.BeforeSuite;  import org.testng.annotations.BeforeTest;  import org.testng.annotations.Test;  public class AnnotationDemo29 {      @BeforeSuite  public void beforeSuite()  {  System.out.println("Before Suite method......");  }    @AfterSuite  public void afterSuite()  {  System.out.println("After suite method........");  }      @BeforeTest  public void beforeTest()  {  System.out.println("Before Test method......");  }    @AfterTest  public void afterTest()  {  System.out.println("After Test method......");  }      @BeforeClass  public void beforeClass()  {  System.out.println("BeforeClass");  }    @AfterClass  public void afterClass()  {  System.out.println("AfterClass");  }      @BeforeMethod  public void beforeMethod()  {  System.out.println("BeforeMethod....");  }    @AfterMethod  public void afterMethod()  {  System.out.println("AfterMethod....");  }    @Test  public void test1()  {  System.out.println("this is test1.....");  }    @Test  public void test2()  {  System.out.println("this is test2.....");  }    @Test  public void test3()  {  System.out.println("this is test3.....");  }    @Test  public void test4()  {  System.out.println("this is test4.....");  }    } | Output |

session – 30

Test annotations Attributes : -

* TestNG m test case by default alphabetically run hote hai agr priority ko specify nhi kiya ho

AnnotationAttributes ( description, enabled, dependsOnMethods, priority, timeOut & groups)

**description syntax :-**

@Test(description = "this is testcase1")

**public** **void** testCase1()

{

System.***out***.println("Mobile Login TestCase.");

}

**timeOut attribute syntax :-** agr koi test case execute hone m bht jyda time le rha hai uski wajha se dusre test case ka execution fail ho rha ho jb timeOut use krte hai

**example** – esme time hume 200 ms set kiya agr esse jyda hoga to ye fail ho jayega program ke ander humne execution time 400ms liye esliye program start se pahle he fail hojayega.

Program =>

@Test(timeOut = 200)

**public** **void** testCase2()

{

**try** {

Thread.*sleep*(400);

}

**catch**(Exception e)

{

e.printStackTrace();

}

System.***out***.println("Web Login TestCase.");

}

**priority attribute syntax :-** test case ki priority ko set krne ke liye eska use krte hai agr 3 test case m se ek ki priority nhi set ki to uski by default **zero** hogi or wo sbse pahle run hoga.

Program =>

@Test(priority = 1)

**public** **void** testCase1()

{

System.***out***.println("Mobile Login TestCase.");

}

**dependsOnMethods attribute syntax –** jb ek test case ka execution dusre test case ke execution pr depend krta ho tb eska use krte hai , exp- agr test1 depend krta hai test2 pr to pahle test2 execute hoga or agr ye fail ho gya to test1 execute he nhi hoga pass hoga jb he execute hoga.

Program =>

@Test(dependsOnMethods = {"testCase2"}) // if method depends two methods {“method1”,”method2”}

**public** **void** testCase1()

{

System.***out***.println("Mobile Login TestCase.");

}

// @Test(timeOut = 200)

// @Test(priority = 2)

@Test

**public** **void** testCase2()

{

**try** {

Thread.*sleep*(400);

}

**catch**(Exception e)

{

e.printStackTrace();

}

System.***out***.println("Web Login TestCase.");

}

**groups attribute syntax :-** eska use similar functionality wale test case ko ek sath group krne ke liye krte hai –

**package** TestNGdemo;

**import** org.testng.annotations.Test;

**public** **class** AnnotationAttributes30two {

@Test(groups = "software company")

**public** **void** infosys()

{

System.***out***.println("Infosys");

}

@Test(groups = "software company")

**public** **void** wipro()

{

System.***out***.println("wipro");

}

@Test(groups = "automobile")

**public** **void** tata()

{

System.***out***.println("tata");

}

@Test(groups = "automobile")

**public** **void** Maruti()

{

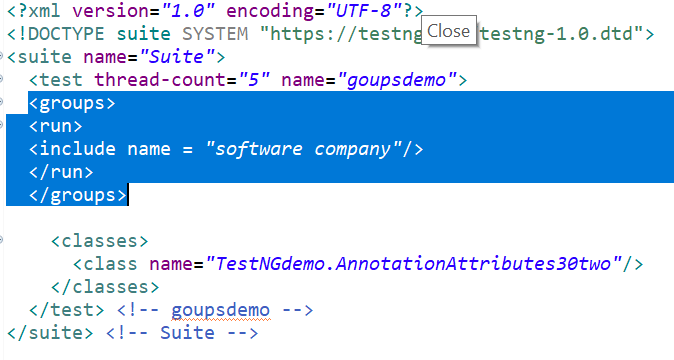
System.***out***.println("Maruti");

}

}

esko execute krne ke liye steps follow krne honge =>

* Testng.xml file m test tag se niche ek <groups> tag create krenge fir groups tag ke anger ek <run> rag create krnege fir es run tag ke ander jis test case ko include krna hai usko <include> tag m likhenge



Session – 31

TestNG Parameters :- TestNG are the arguments that we pass to the test methods. There are two ways through which we can pass the parameters to the test methods.

* **TestNG Parameters ( with XML file )**
* **TestNG DataProviders**

Syntax :-

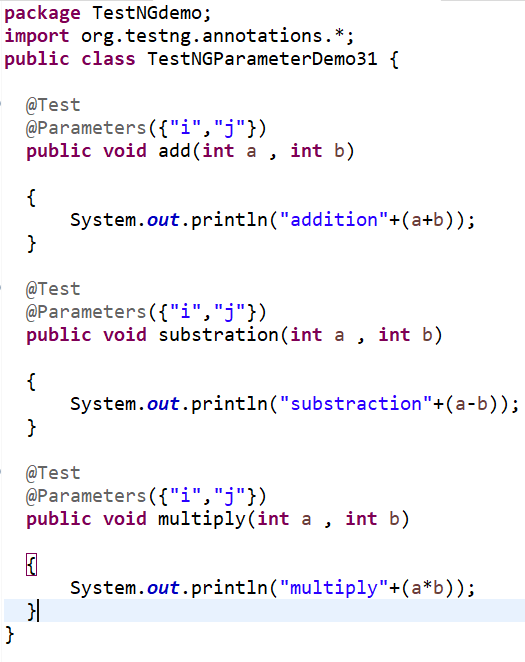
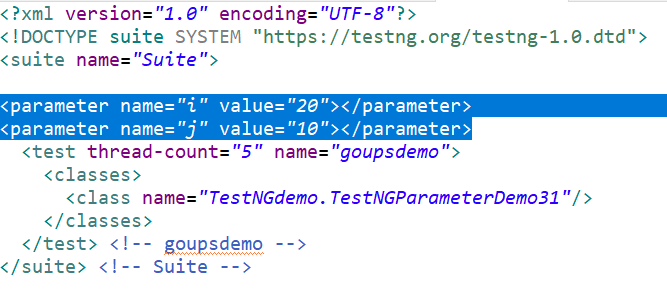
@Parameters({“parameter name”})

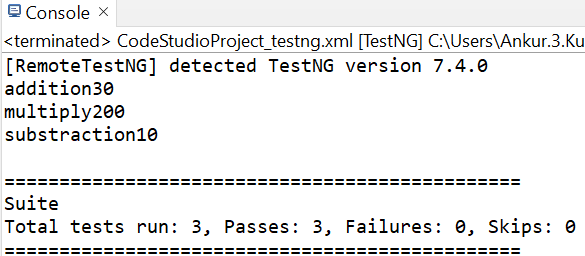
@Parameters({“param1”,”param2”,param3”})

**Steps-**

Parameter tag “text” tag se bahar create krenge agr same parameter value class ke sb test case ko pass krni hai , or agr specific value deni hai to “text” tag ke ander create krenge.

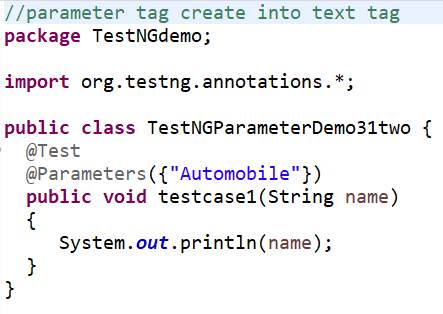
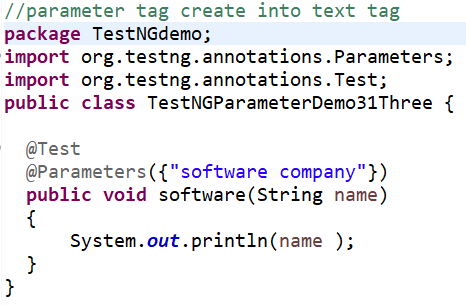
Program- parameter tag in TestNG



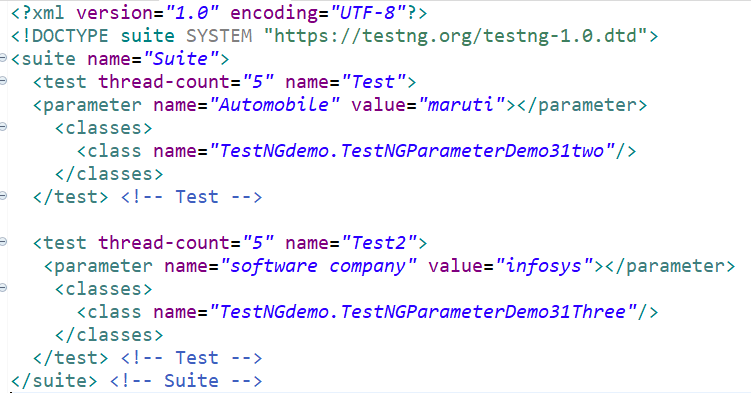
Output -   


**Example 2 –**

**Two different class ke liye parameter in xml file**

** **

**Parameter tag outside the “text” tag**

****

**Practical demo example =>**

**Test data:-**

Keyword – 1) selenium

2) Java

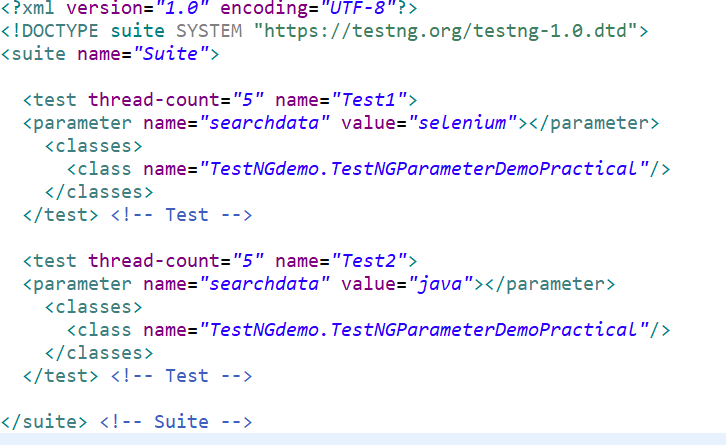
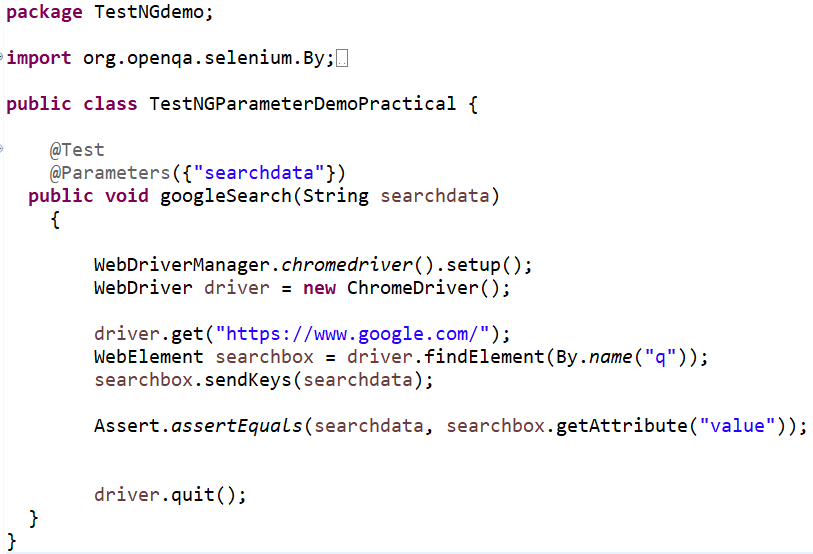
**Steps :-** 1. launch the browser and open www.google.com

2. Add the first keyword as input in the search box

3. verify the input value on UI to be same as from test data

4. Repeat the above two steps for the other 2 keywords

Program :- testng.xml



Program 2 baar run hoga qki 2 parameter pass kiye hai

Realtime example of parameter :-

1. run test case on multiple browser(esme browser parametrized krenge),
2. kisi URL pe login krna hai multiple username of password se to username of password ko parameterized kr skte hai,
3. base url ko bhi parametrized kr skte hai.

Session - 32

Data Provide :- similar to TestNG Parameters, DataProviders are a means to pass data to test methods in TestNG. Using DataProvider in TestNG, we can easily inject multiple values into the same test case. It comes inbuilt in TestNG and is popularly used in data-driven framework.

**Syntax -**

**@DataProvider(name=”name of the data provider”)**

public Object[][] dataProviderFunc()

{

return new Object[][] {value }

}

* The DataProvider annotation has a single attribute called name, which you can select as per your convenience.
* DataProviders are separate methods used in test functions, which means that this annotation is not used on test functions like the testNG parameters.
* The DataProvider method returns a 2D list of objects.
* In case you do not define a name for the DataProvider, the DataProvider method name is considered its default name. So, the name of the DataProvider calls the DataProvider method.

**Test Scenario**

1. launch the browser to open [www.google.com](http://www.google.com)
2. search the first keyword combination.
3. repeat the steps 2 for the other 2 keywords combination.

Search key words

|  |  |  |
| --- | --- | --- |
| s.n | Country | Monument |
|  | India | Qutub minar |
|  | Agra | Taj mahal |
|  | Hyderabad | charminar |

test method annotation -

@Test(dataProvider = "searchDataSet")

data provider method annotation -

@DataProvider(name="searchDataSet")

* agar data provider or test method ek class m ho to jaisa upr syntax hai annotation ka wo likha jayega
* or agr data provider method alag class m hai to class ka naam bhi add krna hoga

test method -

@Test(dataProvider = "searchDataSet", dataProviderClass=Class\_ka\_naam.class)

data provider method -

@DataProvider(name="searchDataSet")

**package** TestNGdemo;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.testng.annotations.DataProvider;

**import** org.testng.annotations.Test;

**import** io.github.bonigarcia.wdm.WebDriverManager;

**public** **class** DataProvider32

{

// 1. india qutubminar

// 2. agra taj mahal

// 3. hyderabad charminar

// es data set m 3 rows or 2 column hai

@DataProvider(name="searchDataSet")

**public** Object[][] searchData()

{

Object[][] searchKeyWord = **new** Object[3][2];

searchKeyWord[0][0] ="india";

searchKeyWord[0][1] ="Qutub Minar";

searchKeyWord[1][0] ="Agra";

searchKeyWord[1][1] ="Taj Mahal";

searchKeyWord[2][0] ="Hyderabad";

searchKeyWord[2][1] ="charminar";

**return** searchKeyWord;

}

@Test(dataProvider = "searchDataSet")

**public** **void** TestCaseGoogleSearch(String country, String monument)

{

//Launch chrome browser

WebDriverManager.*chromedriver*().setup();

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

//open url

driver.get("https://www.google.com/");

WebElement searchbox = driver.findElement(By.*xpath*("//input[@title='Search']"));

//enter key combination of country and monument

searchbox.sendKeys(country + " " + monument);

// click on search button

driver.findElement(By.*name*("btnK")).submit();

driver.quit();

}

}

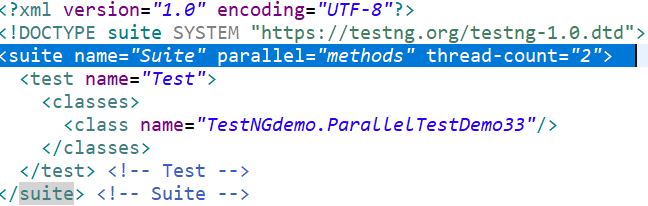
Session – 33

What is Paallel Testing :- eske through test case ka parallel execution kr skte hai esse time bachta hai , eske perform krne ke liye testng m parallel attribute ka use krte hai

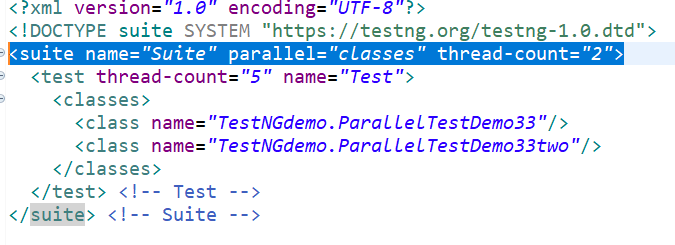
We can execute :-

* all the methods with @Test annotation will execute parallel
* All the test cases inside a java class will run parallel methods
* All the test cases inside <test> tag of Testing xml file will run parallel

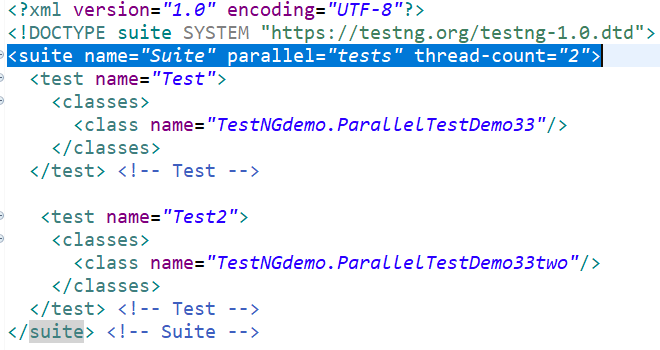
Aagar ek he class hai or uske he method parallel execute krne hai to suite tag m parallel attribute lenge or usme “methods” value lenge qki class ke method ko parallel execute krna hai or thread-count=”2” esliye liya h qki qki humko 2 thread chahiye qki 2 method hai-



Agar 2 class ko parallel execute krna ho to –



Agar Test Suite ko parallel execute krna ho to –



Session – 34

What is Listeners in TestNG :- based on result of the test cases if we have take any action so we use listeners to perform action. [ex. report customised krni ho ya log ko customised krna hai]

Listners is a interface in Testng.

**Type of Listeners in TestNG :-**

1. IAnnotationTransformer
2. IAnnotationTransformer2
3. IConfigurable
4. IConfigurationListener
5. IExecutionListener
6. IHookable
7. IInvokedMethodListener
8. IInvokedMethodListner2
9. IMethodInterceptor
10. IReporter
11. ISuiteListener
12. ITestListner

Methods of iTestListners in TestNG :-

* **onTestStart():** An on TestStart() is invoked only when any test method gets started.
* **onTestSuccess():** An onTestSuccess() method is executed on the success of a test method.
* **onTestFailure():** An on TestFailure() method is invoked when test method fails.
* **onTestSkipped():** An onTestSkipped() run only when any test method has been skipped.
* **onStart():** An onStart() method is executed on the start of any test method.
* **onFinish():** An onFinish() is invoked when any test case finishes its execution.
* **onTestFailedButWithinSuccessPercentage():** This method is invoked each time when the test method fails but within success percentage.

listener ko implement ke liye listner ki ek class bnayenge usme ITestListner ko impliments krnege or uske method ko override krenge es class m override ke liye “ctrl” press krke interface pr click kreng jisse uske implementation show jayegi usme jo methods humare use ke honge copy kr lenge or file m past kr denge or interface m jo methods hote hai sb ko address krna jaruri hai chahe use ho ya na ho

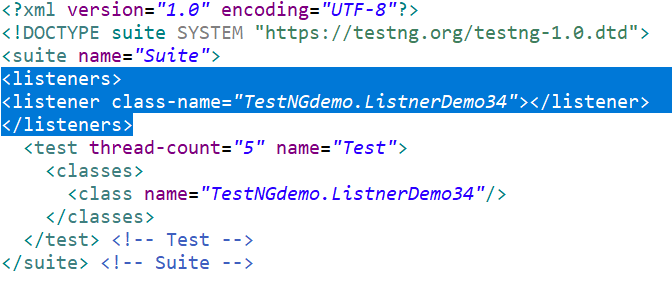
ab test class or Listner class dono ko integrate krna hai uske two ways hai –

* Test class m annotation use krenge (class se pahle create krnege)

syntax :- @Listeners(packagename.classname.class)

@Listeners(TestNGdemo.ListenerClass34.**class**)

* testng.xml file se, agr humare pass bohot sare class honge to sbke samen annotation add krna hoga jbki testng.xml file m ek Listener tag ka use krne multiple classes ko listener class se associate kr skte hai,
* “listeners” tag suite tag se niche create krenge fir eske ander “listener” tag create krnege fir usme class ka naam denge with package-



yha listner class ka naam aayega

Session – 35

Assertion in TestNG :- Assertions in TestNG are used for **validating** the test methods and to **verify that the expected result and the actual result matched or not.**

Types of Assertions

1. Hard Assertion
2. Soft Assertion

**Hard Assertion**

Hard Assertion is an Assertion that **immediately** throws the **AssertException** when the test case is failed

A Hard Assertion contains the following methods

|  |  |  |
| --- | --- | --- |
| S.NO | Method Name | Syntax |
|  | AssertEquals | Assert.assertEquals(actual,expected,message) |
|  | AssertNotEquals | AssertNotEquals(actual,expected,message) |
|  | AssertTrue | Assert.AssertTrue(condition) |
|  | AssertFalse | Assert.AssertFalse(condition) |

**Soft Assertion**

Soft Assert **does not throw an Exception immediately when the assertion fails**, collects them and carries out with the next validation. This accumulates the errors in each @Test execution.

To use **testng** soft assertion, you have to use **testng SoftAssert** class

Test Scenario –

1. launch chrome browser
2. open url –<https://testautomationpractice.blogspot.com/>
3. verify title of the webpage
4. verify the presence of the Wikipedia icon on web page
5. verify the presence of Wikipedia search button web page

* esme hard assert use hai to method 1 m expected result alag liya hai humne to wo fail hoga eski wajha se aage ke execution stop ho jayega -

|  |
| --- |
| **package** TestNGdemo;  **import** **static** org.testng.Assert.*assertTrue*;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.WebElement;  **import** org.openqa.selenium.chrome.ChromeDriver;  **import** org.testng.Assert;  **import** org.testng.annotations.Test;  **import** io.github.bonigarcia.wdm.WebDriverManager;  **public** **class** AssertionDemo35 {    @Test  **public** **void** testMethod() {    //launch chrome browser  WebDriverManager.*chromedriver*().setup();  WebDriver driver = **new** ChromeDriver();    driver.get("https://testautomationpractice.blogspot.com/");    System.***out***.println("verifying titile...");  String expectedTitle = "Automation Testing Practice1"; //1 khud likha fail krne ke liye  String actualtitle = driver.getTitle();  Assert.*assertEquals*(actualtitle, expectedTitle,"Title verifying");    System.***out***.println("verifying presence of wikipedia icon...");    WebElement icon = driver.findElement(By.*xpath*("//img[@class='wikipedia-icon']"));  Assert.*assertTrue*(icon.isDisplayed());      //wikipedia search button  System.***out***.println("Verifying search icon");    WebElement searchIcon = driver.findElement(By.*xpath*("//input[@class='wikipedia-search-button']"));  Assert.*assertTrue*(searchIcon.isDisplayed());    driver.quit();    }  } |

using SoftAssert – (es class ka object crate krnege fir use krnege)

**Syntax =** SoftAssert softverify = new SoftAssert();

|  |
| --- |
| **package** TestNGdemo;  **import** **static** org.testng.Assert.*assertTrue*;  **import** org.openqa.selenium.By;  **import** org.openqa.selenium.WebDriver;  **import** org.openqa.selenium.WebElement;  **import** org.openqa.selenium.chrome.ChromeDriver;  **import** org.testng.Assert;  **import** org.testng.annotations.Test;  **import** org.testng.asserts.SoftAssert;  **import** io.github.bonigarcia.wdm.WebDriverManager;  **public** **class** AssertionDemo35 {    @Test  **public** **void** testMethod() {    //launch chrome browser  WebDriverManager.*chromedriver*().setup();  WebDriver driver = **new** ChromeDriver();    SoftAssert softVerify = **new** SoftAssert();    driver.get("https://testautomationpractice.blogspot.com/");    System.***out***.println("verifying titile...");  String expectedTitle = "Automation Testing Practice1";  String actualtitle = driver.getTitle();  softVerify.assertEquals(actualtitle, expectedTitle,"Title verifying");    System.***out***.println("verifying presence of wikipedia icon...");    WebElement icon = driver.findElement(By.*xpath*("//img[@class='wikipedia-icon']"));  softVerify.assertTrue(icon.isDisplayed());      //wikipedia search button  System.***out***.println("Verifying search icon");    WebElement searchIcon = driver.findElement(By.*xpath*("//input[@class='wikipedia-search-button']"));  softVerify.assertTrue(searchIcon.isDisplayed());    driver.quit();    //report all failure messages  softVerify.assertAll(); // assertAll jitne bhi failure msg hai usko report krenga agr report nhi krenge to testcase status pass dikhayega    }  } |

Session – 36

What is Batch Testing :-

* Running **multiple test cases** in a **suite** is called **Batch Testing.**
* A **test suite**  is a **collection of test cases.** Test suites **help in grouping test cases.**
* You can categorize test suites based on functionality, module, environment, or something else.
* In testNG we can use **XML file to perform batch Testing**

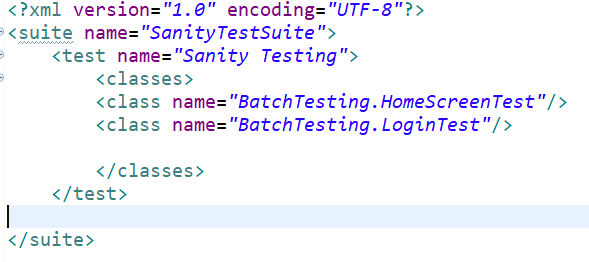
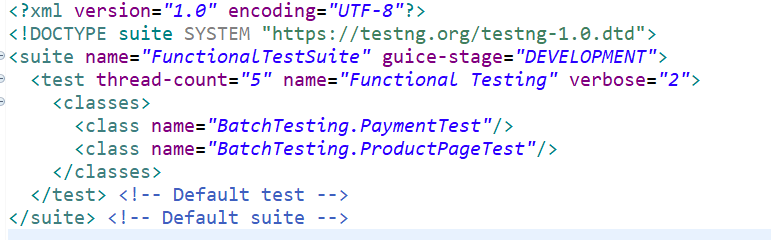
Example of Batch Testing -

|  |  |  |
| --- | --- | --- |
| Suite | Test cases | Test Methods |
| Sanity Test | HomeScreenTest | LaunchApplication() |
|  |  | VerifyTitle() |
|  |  | VerifyLogo() |
|  |  |  |
|  | Login Test | LoginByMobileNumber() |
|  |  | LoginByEmail() |
|  |  |  |
| Functional Test | ProductPageTest | AddProductToWishList() |
|  |  | AddProductToCart() |
|  |  | SelectQuantity() |
|  | Payment Test | CashOnDelivery() |
|  |  | NetBanking() |

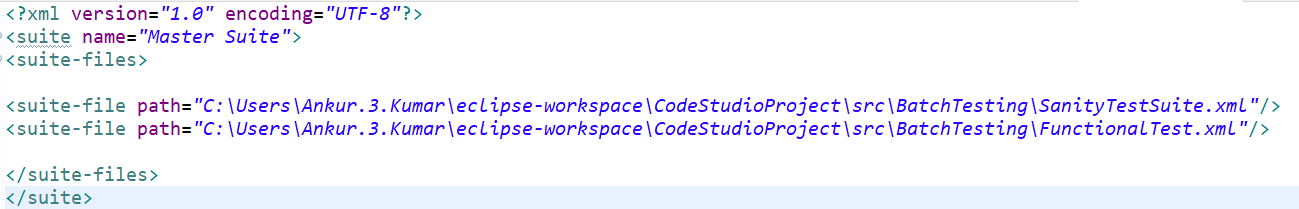
‘’test-output’’ folder m “index.html” file create hoti hai jb bhi koi test case execute krte hai agr batch testing krni ho or alag alag xml file create krni ho to pahle jin test case ki xml file create krni ho unko run kro “index.xml” m se copy krke , us package m xml file create krke paste kr skte hai

multiple XML file kaise ek sath run krte hai :-

1. ek seprate xml file create krenge (example – MasterTestSuite.xml)

senity test xml functional test xml



mastertest xml file

Session – 37

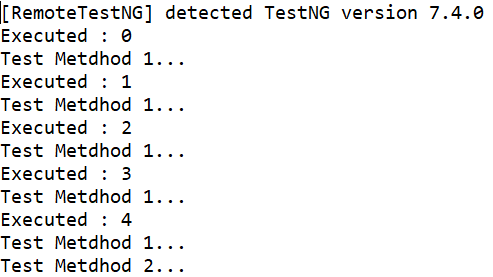
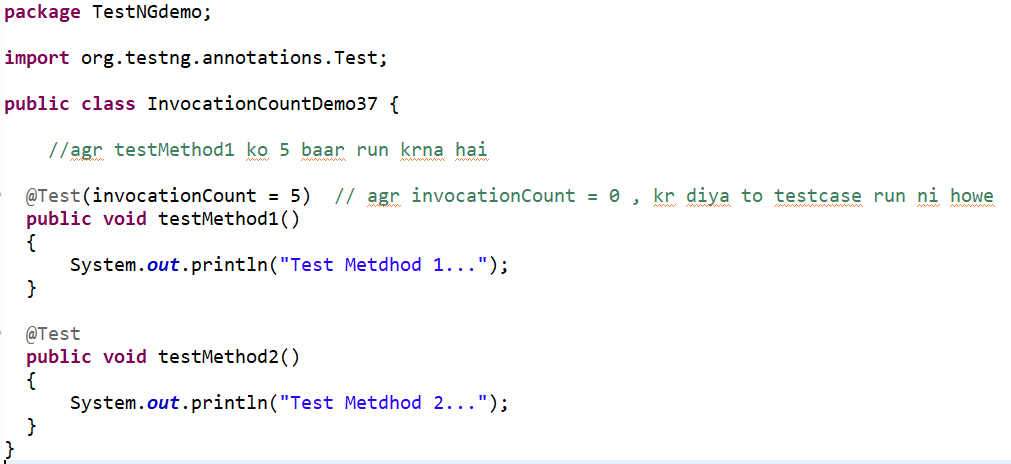
@Test annotation- invocationCount attribute

In TestNG, InvocationCount attribute is used to run single test case multiple time.

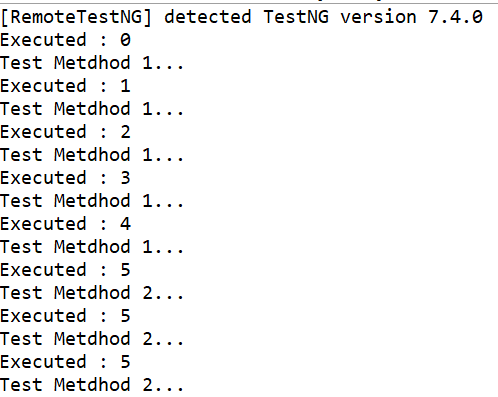
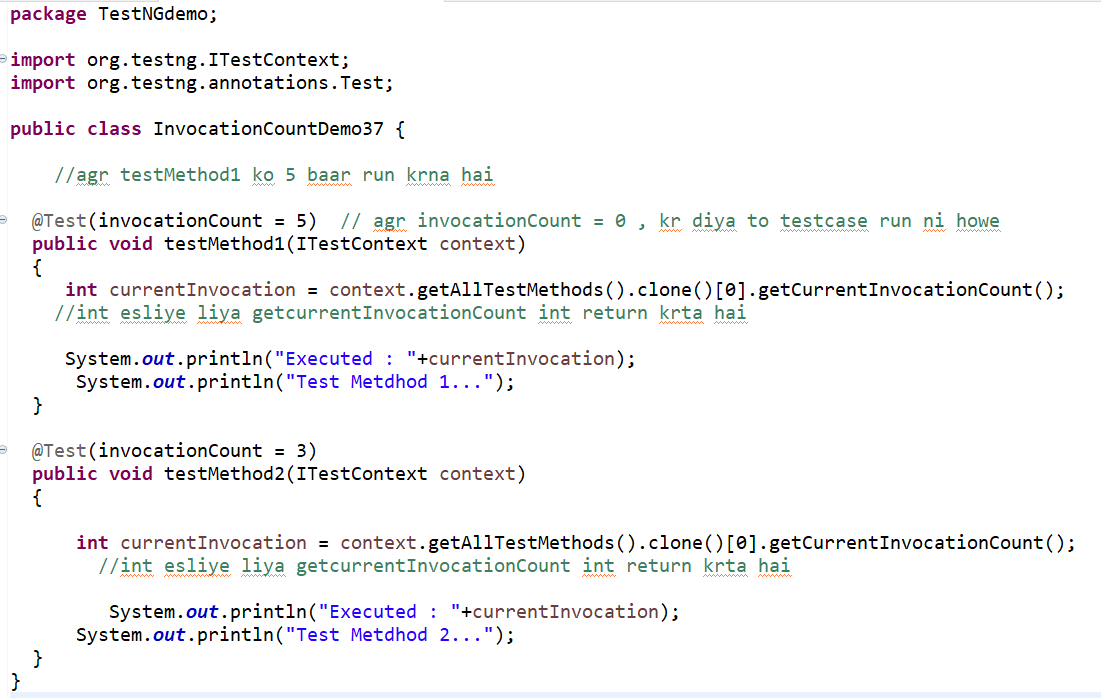
**Syntax :-**

@Test(invocationCount = num)

Where num = number of times you want to run this test method.



agr humko dekhna hai konsa current method konsa run ho rha hai to method m **ITestContext parameter** ka use krte hai



Session – 38

**Log4j2 Logging Framework**

What is Logging:- **Logging** means some way to indicate the **state of the system at runtime**. the log messages have to provide the required information to understand what the **application does internally during runtime.**

What is Log4j :-

Log4j **logging framework** which is written in java. It is an **Open-source logging API for java.**

Configuration of Log4j in eclipse Maven Project :-

1. download **Log4j-core** dependency
2. download **Log4j-api** dependency
3. **Log4j.properties** file or **Log4j2.xml** file

paste these log4j2.properties or log4j2.xml in project home directory.

Log4j2.properties file is a log4j configuration file

which stores entire runtime configuration used by log4j.

location: **src/main/resource** (es naam se folder create krna hoga project m esko right click krke source folder m jake create krte hai es naam se)

1. import log4j package

import org.apache.logging.log4j.\*;

1. Create object of Logger

Logger log = LogManager.getLogger(“ClassName)

<https://logging.apache.org/log4j/2.x/>

**Custom Log Levels**

**Defining Custom Log Levels in Code**

Log4J 2 supports custom log levels. Custom log levels can be defined in code or in configuration. To define a custom log level in code, use the Level.forName() method. This method creates a new level for the specified name. After a log level is defined you can log messages at this level by calling the Logger.log() method and passing the custom log level:

// This creates the "VERBOSE" level if it does not exist yet.

final Level VERBOSE = Level.forName("VERBOSE", 550);

final Logger logger = LogManager.getLogger();

logger.log(VERBOSE, "a verbose message"); // use the custom VERBOSE level

// Create and use a new custom level "DIAG".

logger.log(Level.forName("DIAG", 350), "a diagnostic message");

// Use (don't create) the "DIAG" custom level.

// Only do this \*after\* the custom level is created!

logger.log(Level.getLevel("DIAG"), "another diagnostic message");

// Using an undefined level results in an error: Level.getLevel() returns null,

// and logger.log(null, "message") throws an exception.

logger.log(Level.getLevel("FORGOT\_TO\_DEFINE"), "some message"); // throws exception!

When defining a custom log level, the intLevel parameter (550 and 350 in the example above) determines where the custom level exists in relation to the standard levels built-in to Log4J 2. For reference, the table below shows the intLevel of the built-in log levels.

|  |  |
| --- | --- |
| Standard log levels built-in to Log4J | |
| **Standard Level** | **intLevel** |
| OFF | 0 |
| FATAL | 100 |
| ERROR | 200 |
| WARN | 300 |
| INFO | 400 |
| DEBUG | 500 |
| TRACE | 600 |
| ALL | Integer.MAX\_VALUE |

Selenium – 39

Page Object Model

**Page Object Model (POM)** is a design pattern, popularly used in test automation that creates Object Repository for web UI elements

Under POM

* Each web page is application
* There is **separate class for each web page** to identify web elements of that web page and methods which perform operations on those Web Element

POM Implementation

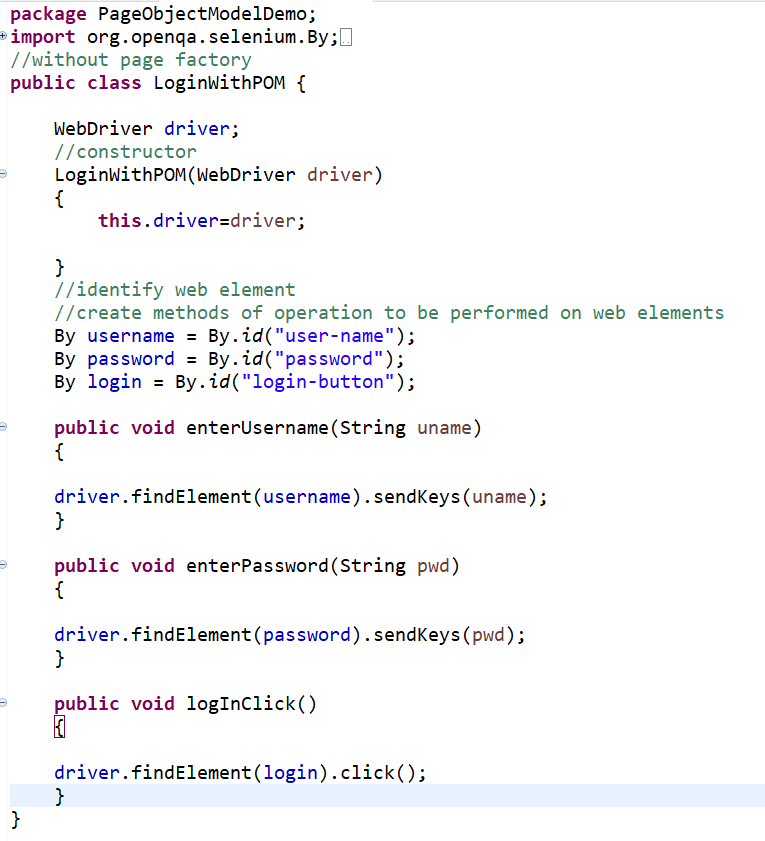
Without Page Factory With Page Factory

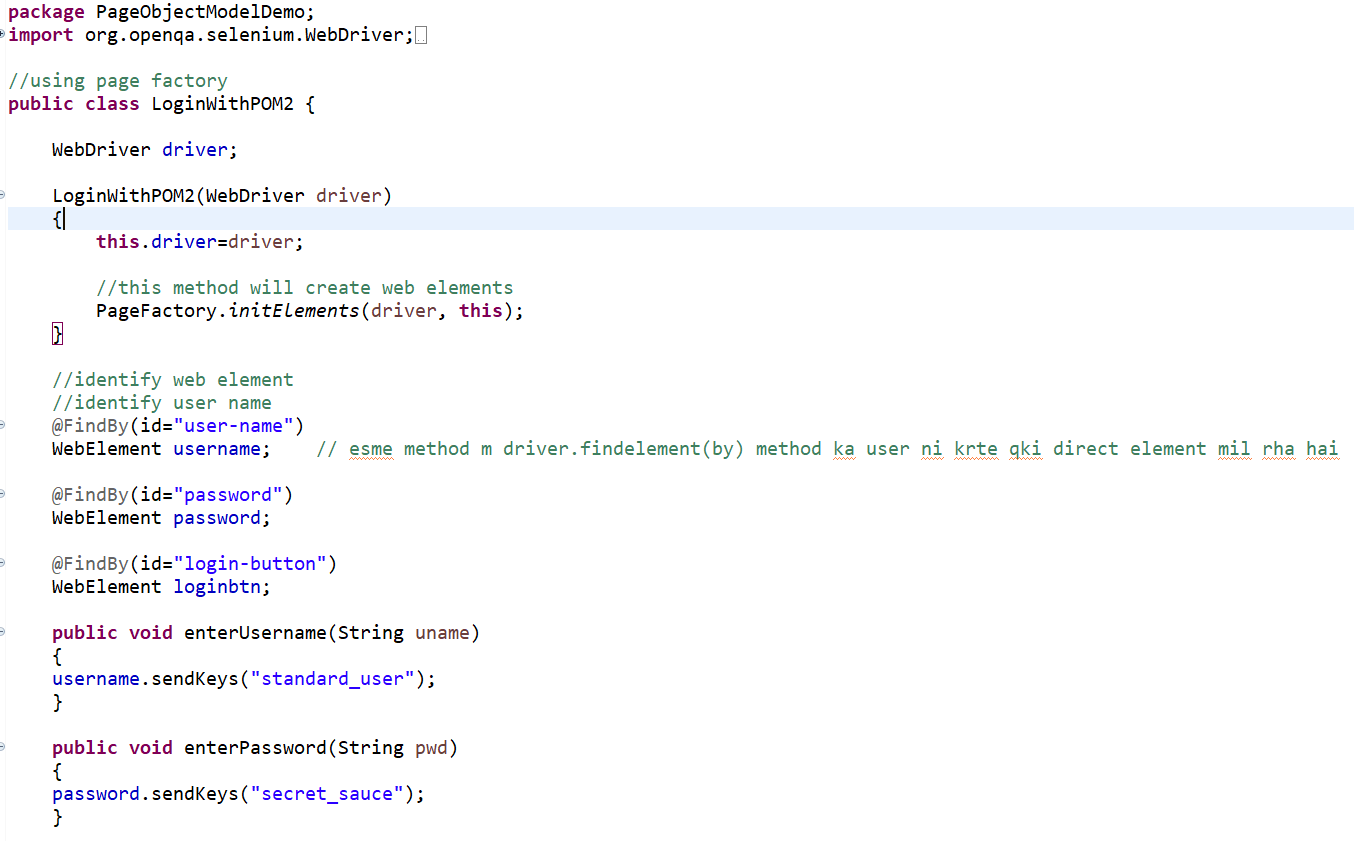
* By().method -> @FindBy annotation
* findEement() -> PageFactory.initElement()

this initElements method will create all WebElements

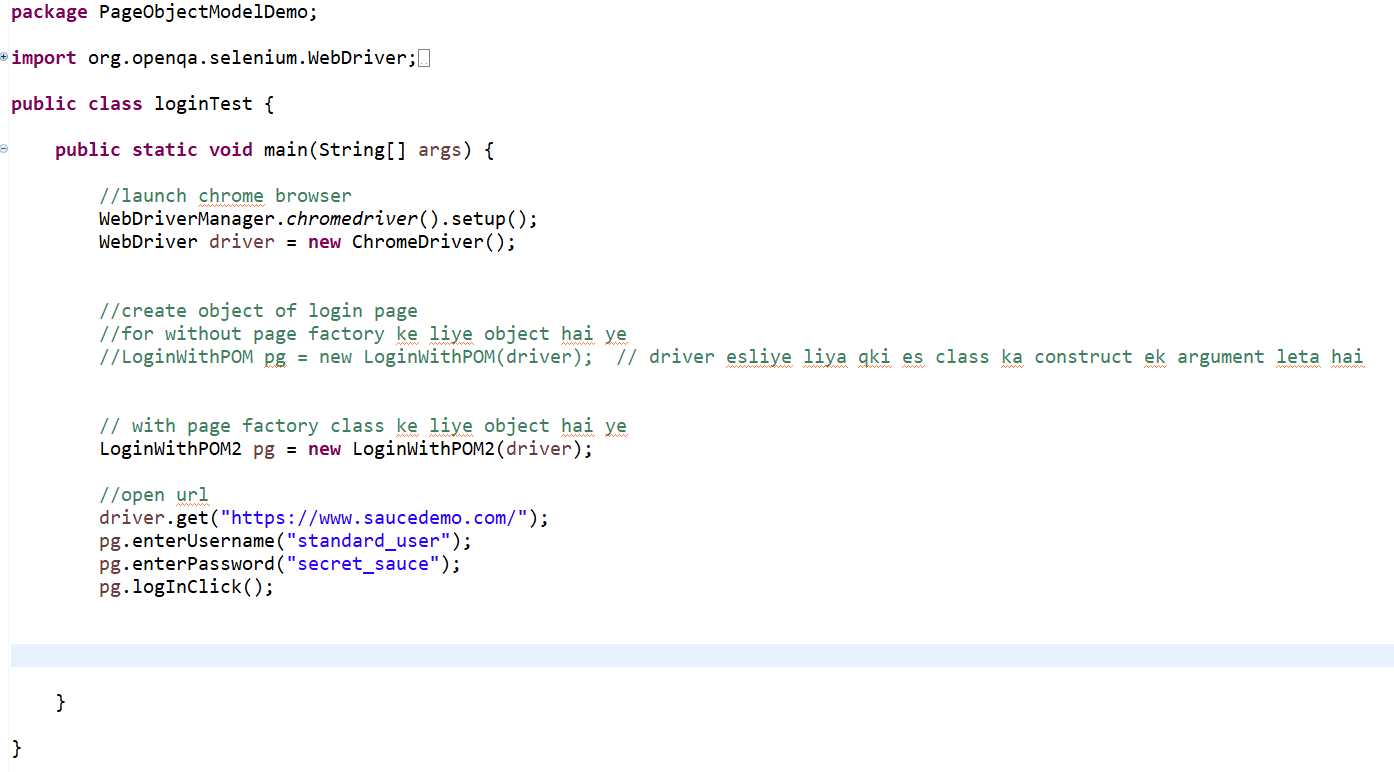
* esme ek class m web element or jo methods perform hone hai we likhte hai or ek class or create krte hai usme Testcases ko write krenge.

eska example codestudioproject2 m hai









Session – 40

Cross Browser Testing (CBT) :- A process to perform tests on multiple browsers.

This is done to know how a website performs on different browsers.

Cross browser testing gives the confidence that the website behaviour is consistent across various browser

eska example codestudioproject2 m milega

**package** CrossBrowserTesting;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.edge.EdgeDriver;

**import** org.testng.Assert;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.BeforeMethod;

**import** org.testng.annotations.Parameters;

**import** org.testng.annotations.Test;

**import** io.github.bonigarcia.wdm.WebDriverManager;

**public** **class** CrossBrowserTestingDemo {

WebDriver driver;

@BeforeMethod @Parameters("browser")

**public** **void** LaunchBrowser(String browser)

{

**switch**(browser.toLowerCase())

{

**case** "chrome":

WebDriverManager.*chromedriver*().setup();

driver = **new** ChromeDriver();

**break**;

**case** "msedge":

WebDriverManager.*edgedriver*().setup();

driver = **new** EdgeDriver();

**break**;

**default**:

driver = **null**;

**break**;

}

}

@Test

**public** **void** verifyBrowser()

{

//open url

driver.get("https://www.google.com/");

String expectedTitle = "Google";

String actualTitle = driver.getTitle();

Assert.*assertEquals*(actualTitle, expectedTitle);

}

@AfterMethod

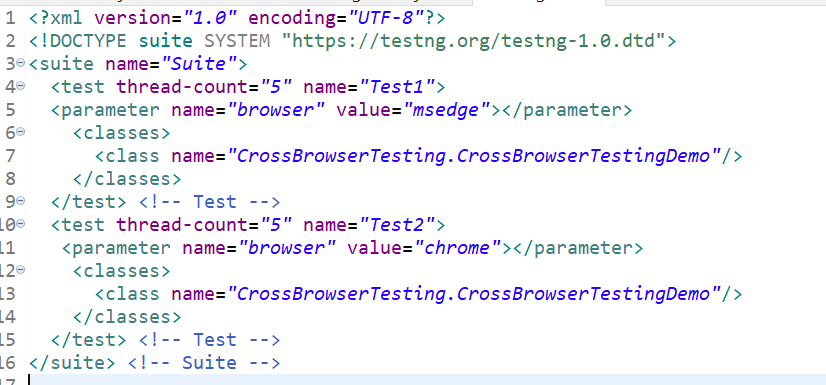
**public** **void** Quit()

{

driver.quit();

}

}



Session – 41

Why we re-run the test case when it fails :-

There are multiple reasons why the test fails

* Due to the network issue
* due to application downtime
* Due to loading issue and etc

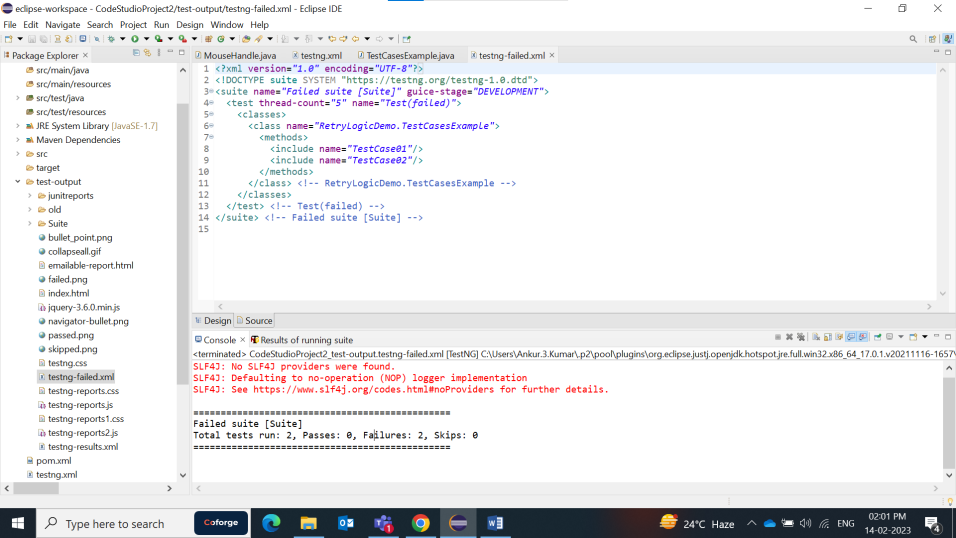
But if the script is failing due to xpath and some valid reason then you have to maintain for re work on your scripts.

How to Re run Failed test cases in Selenium in case of failure?

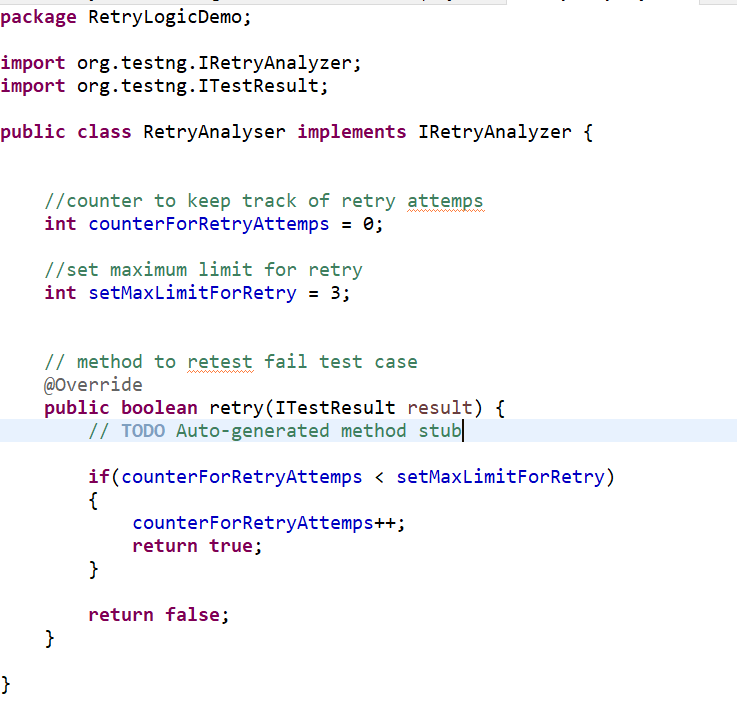
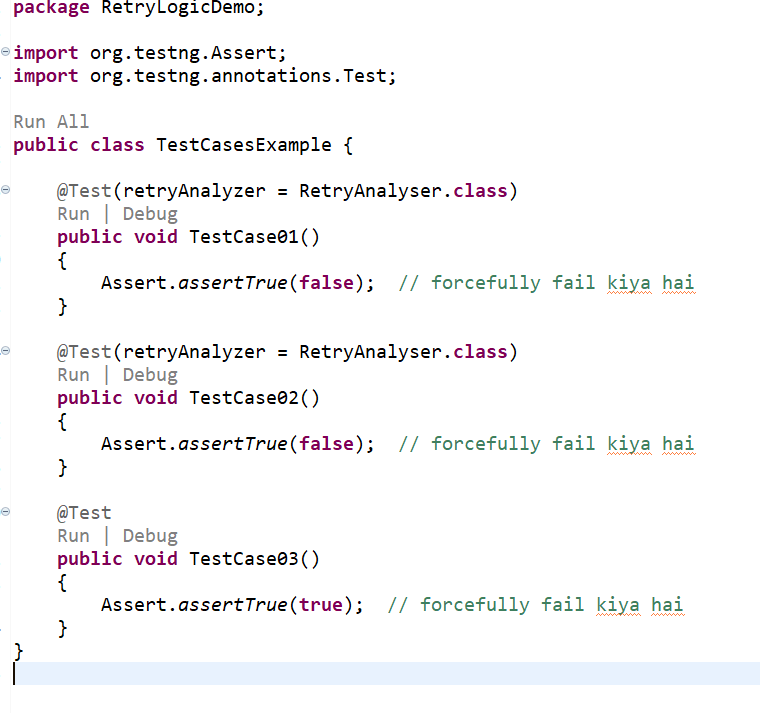
1. Using testing-failed.xml
2. Using IRetryAnalyer interface that is part of TestNG and we need to override retry method.

eska example codestudioproject2 m milega

1. using testng-fail.xml file {ye test-output wale folder m create hogi execution ke baad}



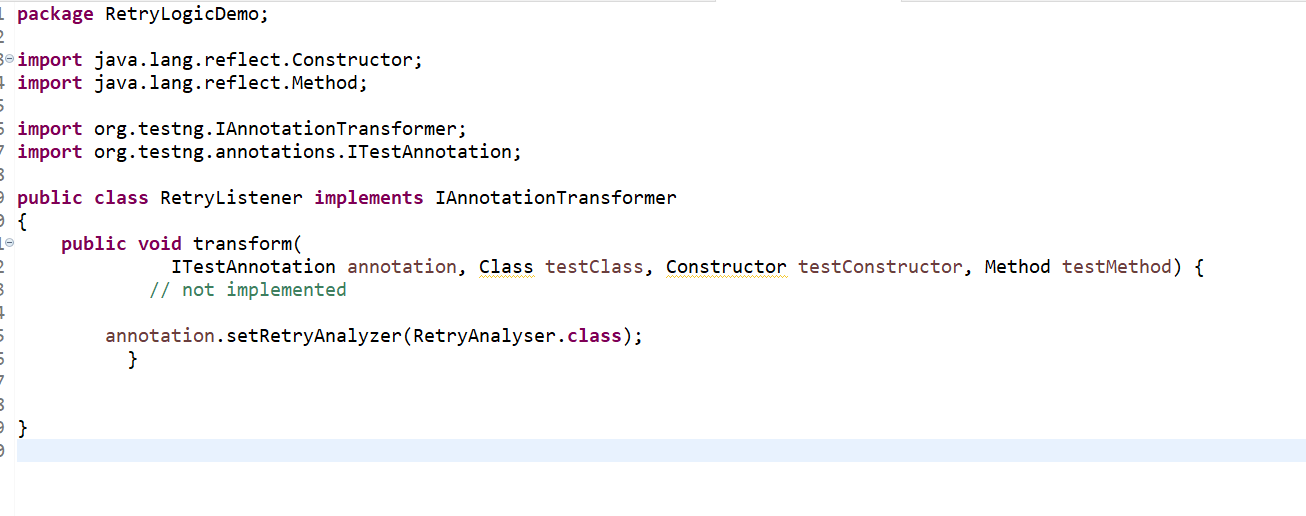
2 – IRetryAnalyser – ek class create krenge usme IRetryAnalyzer interface ko implement krenge and esko Testcase m test tag m @Test(retryAnalyzer = RetryAnalyser.**class**) ese likhenge esme retryAnalyzer class ka naam hai

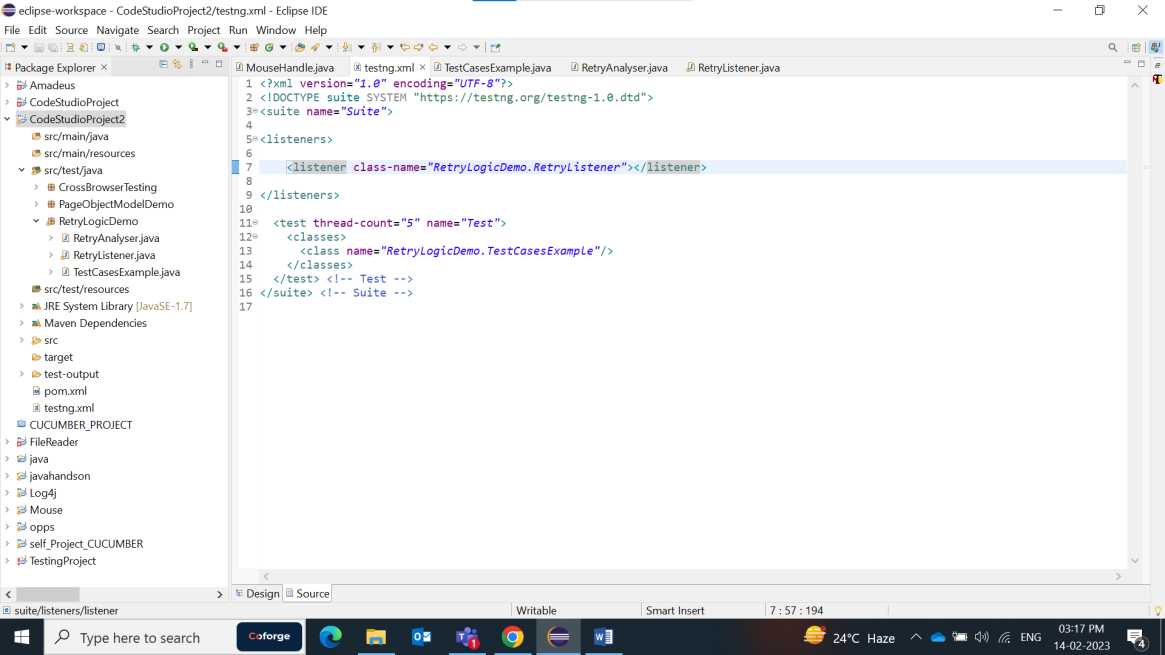
 

ab agr project m bht sare testcase ho to sbko manual test annotation m value pass nhi kr skte bohot jyda ho jayega eske alaga ek interface ka use krnge ye ek listner hai

or wha se test annotation se sb remove kr denge

Note = agr sare test case m Retry functionality nhi apply nhi krna bs limited m krna hai to IAnnotationTransformation ki jarurt nhi hai na he textng.xml m likhne ki jarut hai





Session – 42

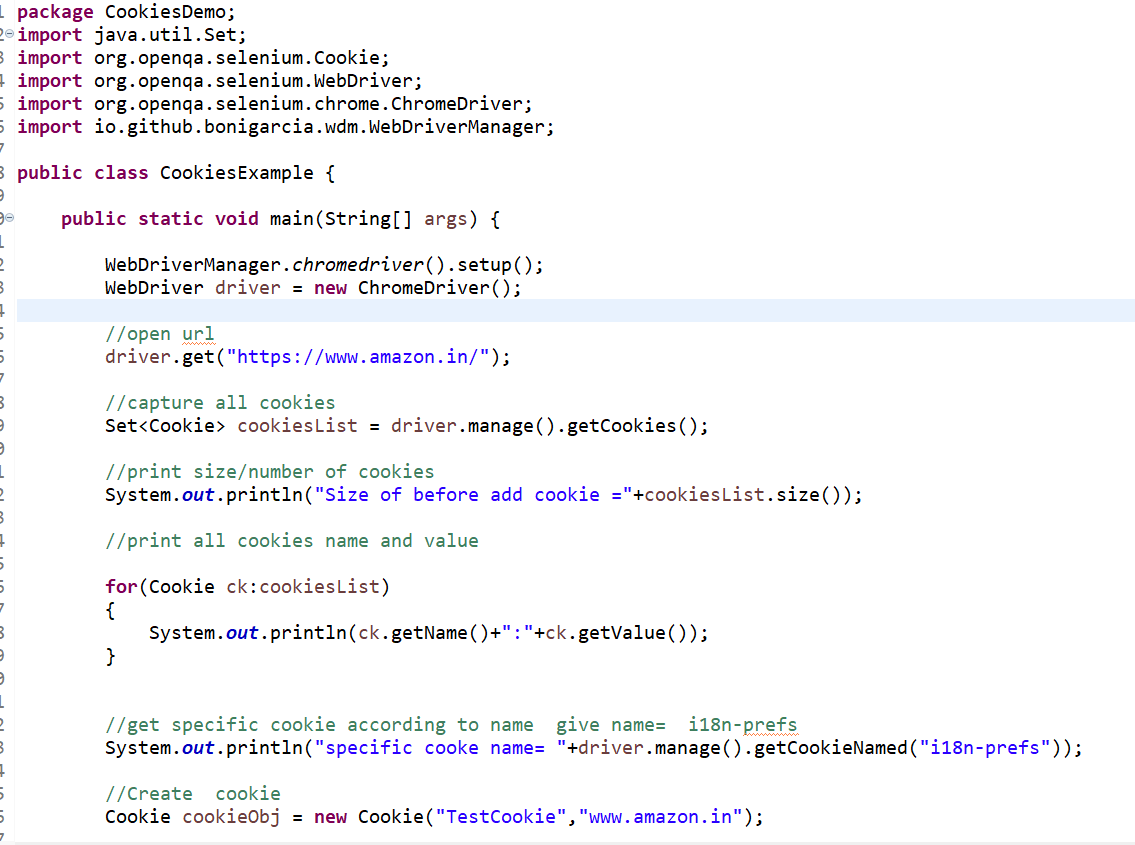
How to Handle Cookies

What is HTTP Cookie :- A **HTTP cookie** is **comprised of information about the user and their preferences.** it is a small piece of data sent **from Web Application** and **stored in Web Browser**, while the user is browsing that website.

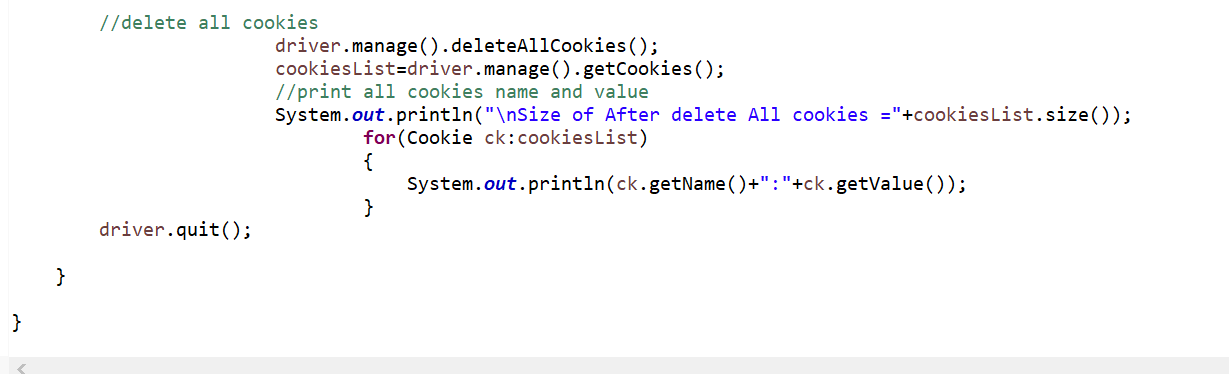
**Method to query and Interact with Cookies :**

1. driver.manage().getCookies(); // return the list of all cookies
2. driver.manage().getCookieNamed(arg0); // Retrun specific cookie according to name
3. driver.manage().addCookie(arg0); // Create and add the cookie
4. driver.manage().deleteCookie(arg0); // Delete specific cookie
5. driver.manage().deleteCookieNamed(arg0); // delete specific cookie according name
6. driver.manage().deleteAllCookies(); // delete all cookies

**Program :-**







Session - 43

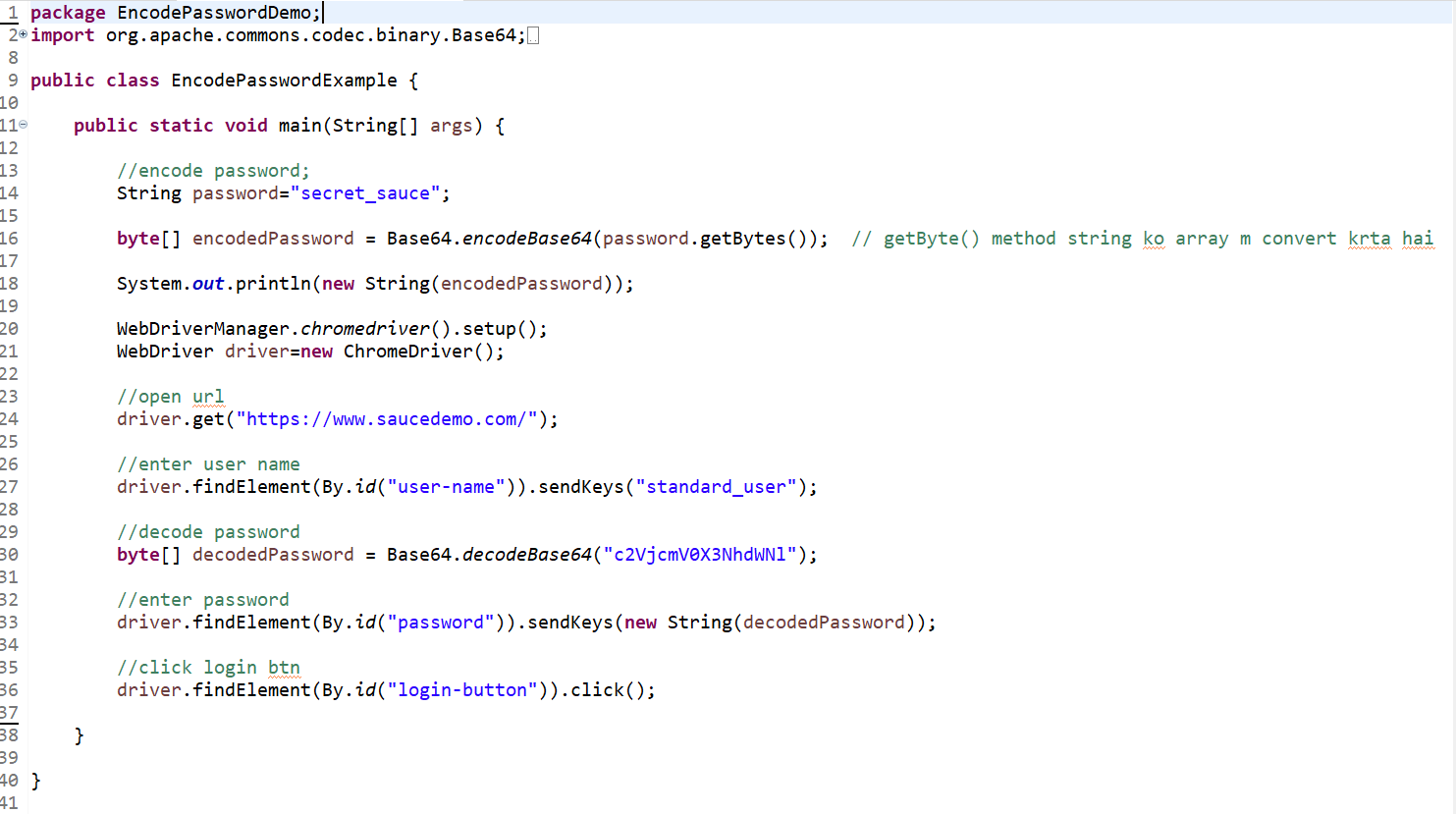
How to Encode Password/Sensitive Data :-

What is Encoding/Encryption :-

**Encryption** is the process of **converting plain text data into an unreadable format** in order to protect unauthorized access.

To Secure our passwords/sensitive data we can use **Base64 encoding scheme in selenium WebDriver.**

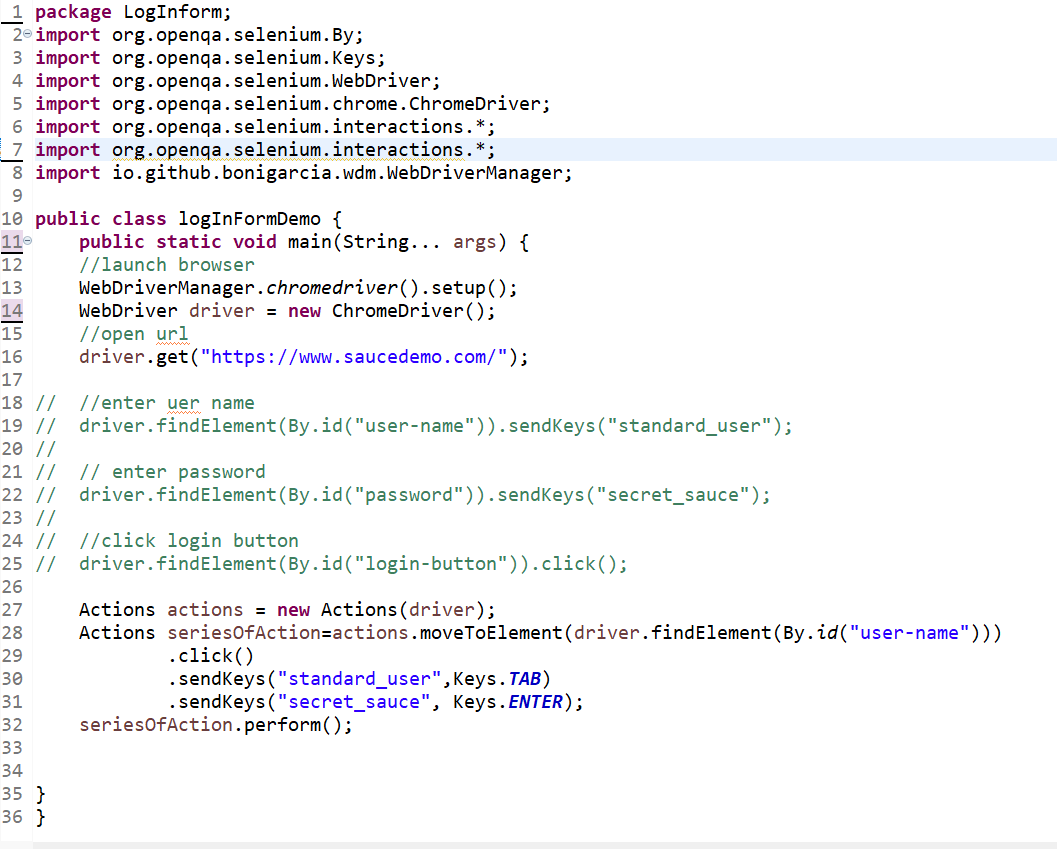
We will **import** this **Base64 class**  in our script to **decode** password/Sensitive data.



Session – 44

How to Fill a Form in a single Statement :-

seriesOfAction se pahle Action likhte hai pr usme error aara tha esliye Actions likha hai



Session – 45 important

Data Driven Testing :-

Data-driven testing is a test automation framework which stores test data in a table or spread spreadsheet format( Eg Excel file)

Using java classes and Apache POI library we can manipulate (read/write) Excel files (both XLS and XLSX file format) in selenium WebDriver.

Steps to read data from excel :-

1. create a maven java project.
2. navigate to maven central repository.
3. search for

you need to copy below dependencies:

1. **poi**
2. **poi-ooxml**
3. click on central tab and copy latest dependencies.
4. paste above dependencies in pom.xml of your maven project and save it.

Note :- As soon as you save it, Maven will build workspace with apache poi dependencies.

**Excel-> workbook-> Sheet-> row-> cell(cell m data milega)**

How to Read Data From Excel File( Apache POI ) :-

**package** DataDrivenTestingDemo;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** org.apache.poi.xssf.usermodel.XSSFCell;

**import** org.apache.poi.xssf.usermodel.XSSFRow;

**import** org.apache.poi.xssf.usermodel.XSSFSheet;

**import** org.apache.poi.xssf.usermodel.XSSFWorkbook;

**public** **class** DataDrivenTestingExample {

**public** **static** **void** main(String[] args) {

XSSFWorkbook ExcelWorkbook=**null**;

XSSFSheet ExcelSheet;

XSSFRow Row;

XSSFCell cell;

FileInputStream inputStream =**null**;

//create an object of File class to open file

File excelFile = **new** File("C:\\Users\\Ankur.3.Kumar\\Desktop\\tutorials\\TestDataFile.xlsx");

//create an object of FileInputStream to read data from file

**try** {

inputStream = **new** FileInputStream(excelFile);

} **catch** (FileNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

//Excel-> workbook-> Sheet-> row-> cell(cell m data milega)[jo upr object hai usko memory allocate krnege]

//create object of XSSFWorkbook to handle excel file

**try** {

ExcelWorkbook = **new** XSSFWorkbook(inputStream);

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

//to access workbook sheet

ExcelSheet = ExcelWorkbook.getSheetAt(0); //qki workbook ke ander sheet hoti hai esliye esse get ki

//get total row count

**int** totalRows = ExcelSheet.getLastRowNum() + 1;

//get total cells/column in a row

**int** totalCells = ExcelSheet.getRow(0).getLastCellNum(); //ye bydefault +1 krke deta hai

**for**(**int** currentRow = 0; currentRow<totalRows; currentRow++) // to read row

{

**for**(**int** currentCell=0 ; currentCell<totalCells; currentCell++) //to read cell

{

System.***out***.print(ExcelSheet.getRow(currentRow).getCell(currentCell).toString());

System.***out***.print("\t");

}

System.***out***.println();

}

**try** {

ExcelWorkbook.close();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

Session – 46 project 2 m milega code

Data Driven Framework (TestNG dataProvider with Excel)

TestNG Parameters :- **TestNG parameters** enable us to pass the values only once per execution cycle.

TestNG DataProvider :- DataProvider in TestNG allows us to pass multiple parameters to a single test in a single execution. using DataProviders, we can easily pass multiple values to a test in just one execution cycle;

Session – 47

What is A Headless Browser :-

Headless browser, means a Web Browser without user interface. To elaborate, headless browsers are those which actually access the web page, but the GUI / Screen in hidden from the user.

esko baad m kr lenge---------------------

Session – 48 important

TestNG Reports

**TestNG reports** are the default HTML Reports which are generated once the test cases are executed using TestNG

The TestNG will generate the default report

* index.html (detail report)
* emailable-report.html (summarizes)
* Reporter class – Used to log information in reports

**Note: Selenium web driver** is used for automating the web-application, but it won’t generate any reports.

Reporter Output – index.html m reporter output m agr koi info ya msg ko lock krna hai Reporter class ka use krnege

-jo bhi operation kiye hai reporter class m unko lock krna hai to reporter class ka user krenge

**Reporter Class Syntax :**

Reporter Class of TestNG provides us with four different methods to log information. Isn’t that interesting? Here are those methods:

* Reporter.log(String s);
* Reporter.log(String s, Boolean logToStandardOut);
* Reporter.log(String s, int level);
* Reporter.log(String s , int level, Boolean logToStandardout

1. Reporter.log(String s);

This method logs the string passed into your HTML Report.

1. Reporter.log(String s, Boolean logToStandardOut);

This method logs the string passed into your html report.

Additionally, it also prints the same message on your console if logToStandardout is set to TRUE.

Parameters :

S – The message to be logged

logToStandardOut – Print the message on standard output

1. Reporter.log(String s, int level);

This method logs the string passed into your html report if the current verbosity equals or is greater than the one passed in the parameter.

Parameters :

S – The message to be logged

level – The verbosity of the message to be logged

**What is verbosity level in TestNG?**

The verbose level in TestNG is used to define the amount of logging performed on the console. the verbosity level ranges from 0 to 10, where 10 is the most detailed logging level whereas 0 means minimal logging.

you can set the verbosity levelin your testng.xml

<suite thread-count=”2” name =”TestNGReporterTest” parallel=”classes” verbose=”10”>

1. Reporter.log(String s, int level, Boolean logToStandardOut);

Parameters :

S – The message to be logged

level – The verbosity of the message to be logged

logToStandardOut – Whether to print the message on standard output as well

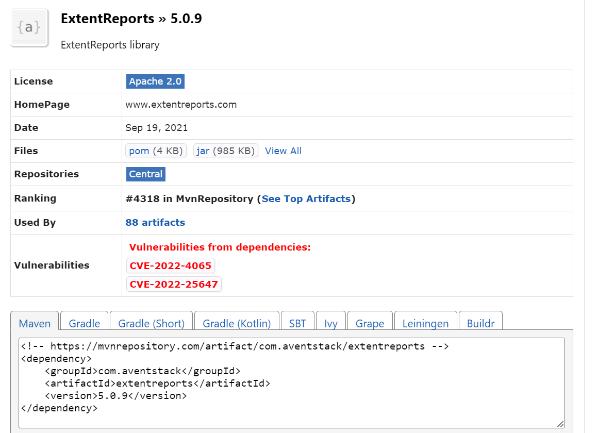


Section – 49 important

Extent Report :

Extent Report is an open-source HTML reporting library useful for test automation. Extent API can produce more **interactive reports,** a **dashboard view, graphical view, capture screenshots at every test step , and emailable reports**

Extent Report Maven Dependency-

  
<!-- https://mvnrepository.com/artifact/com.aventstack/extentreports -->

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

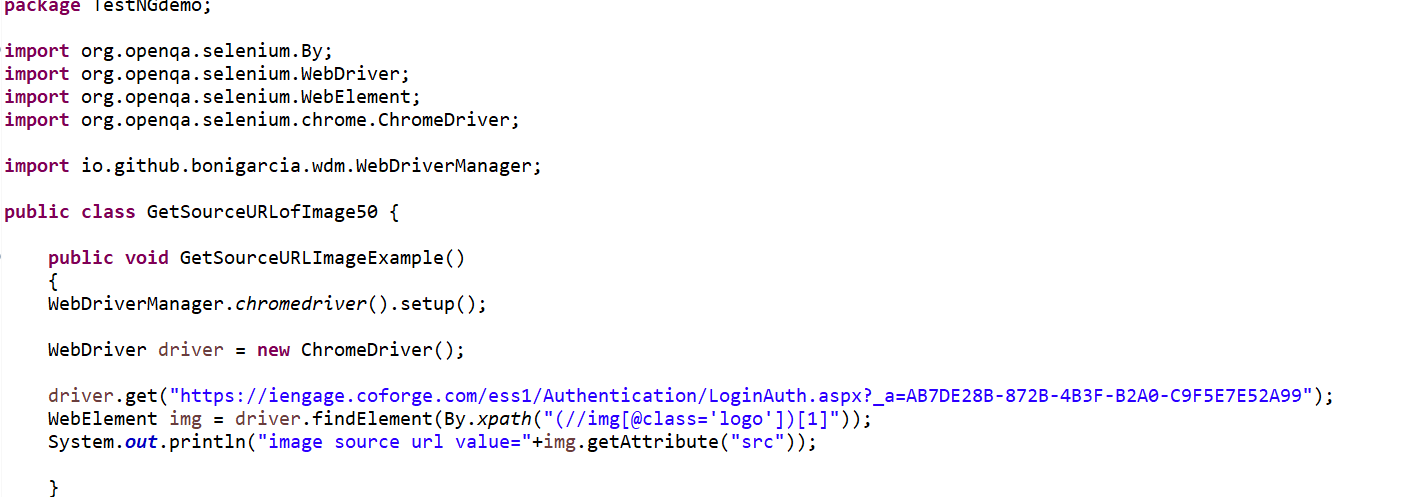
<version>5.0.9</version>

</dependency>

**Note** – Program project 2 m hai

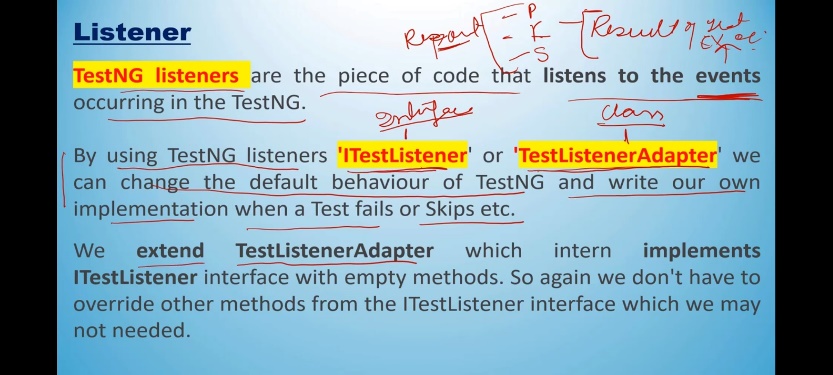
Session – 50

How to get source URL of Image :



Session – 51

Extent Report Using TestNG Listener :-

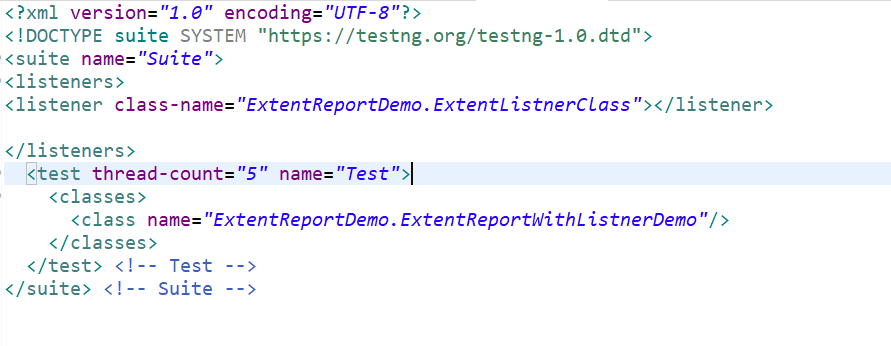




**Extent Report using Test Listener**

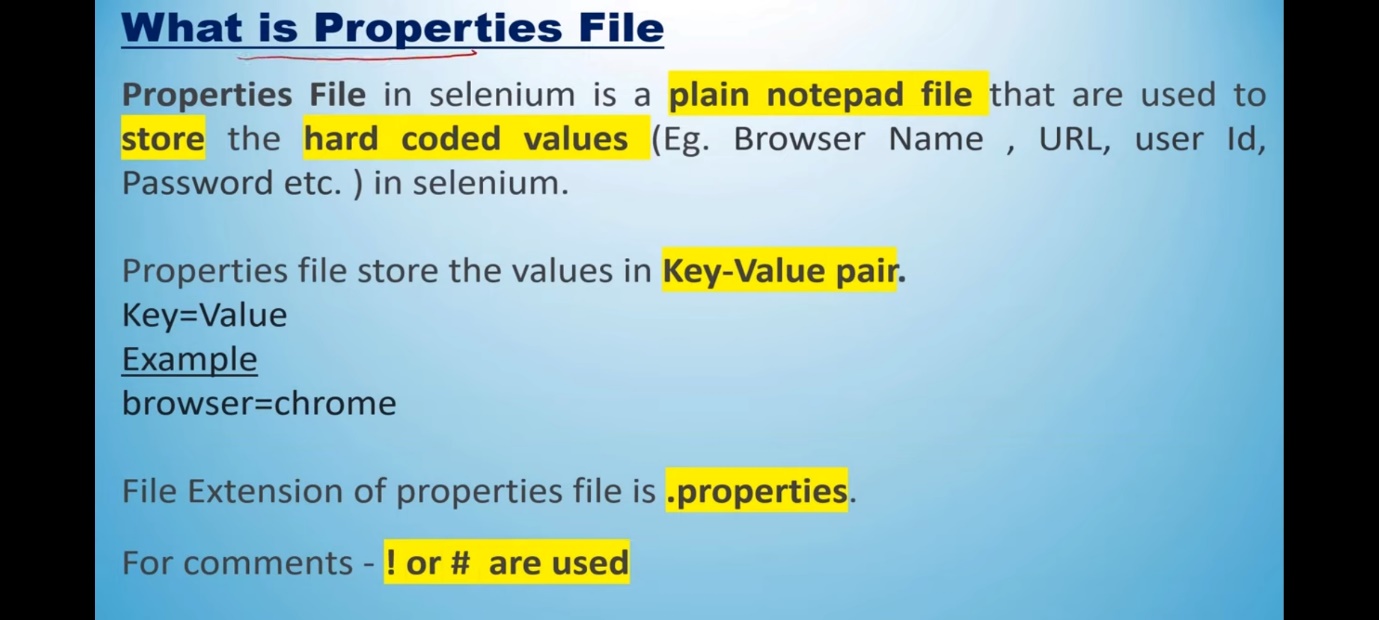
1. create test class for test methods
2. create testng.xml to execute test case
3. create testNG listener class
4. esi class report ko configure krenge
5. or report ko log eske he methods m krenge jo ITestListner ke hai
6. or esko listner class ko testng.xml m bhi configure krni hai jisse run how file

or files project 2 m Extentlistnerdemo krke hai usme hai



Session – 52 important

How to Read Configuration From Properties files



text file ke liye bhi bilkul same krte hai bs file location m jake .properties ki jagha .txt aajata hai