UPES

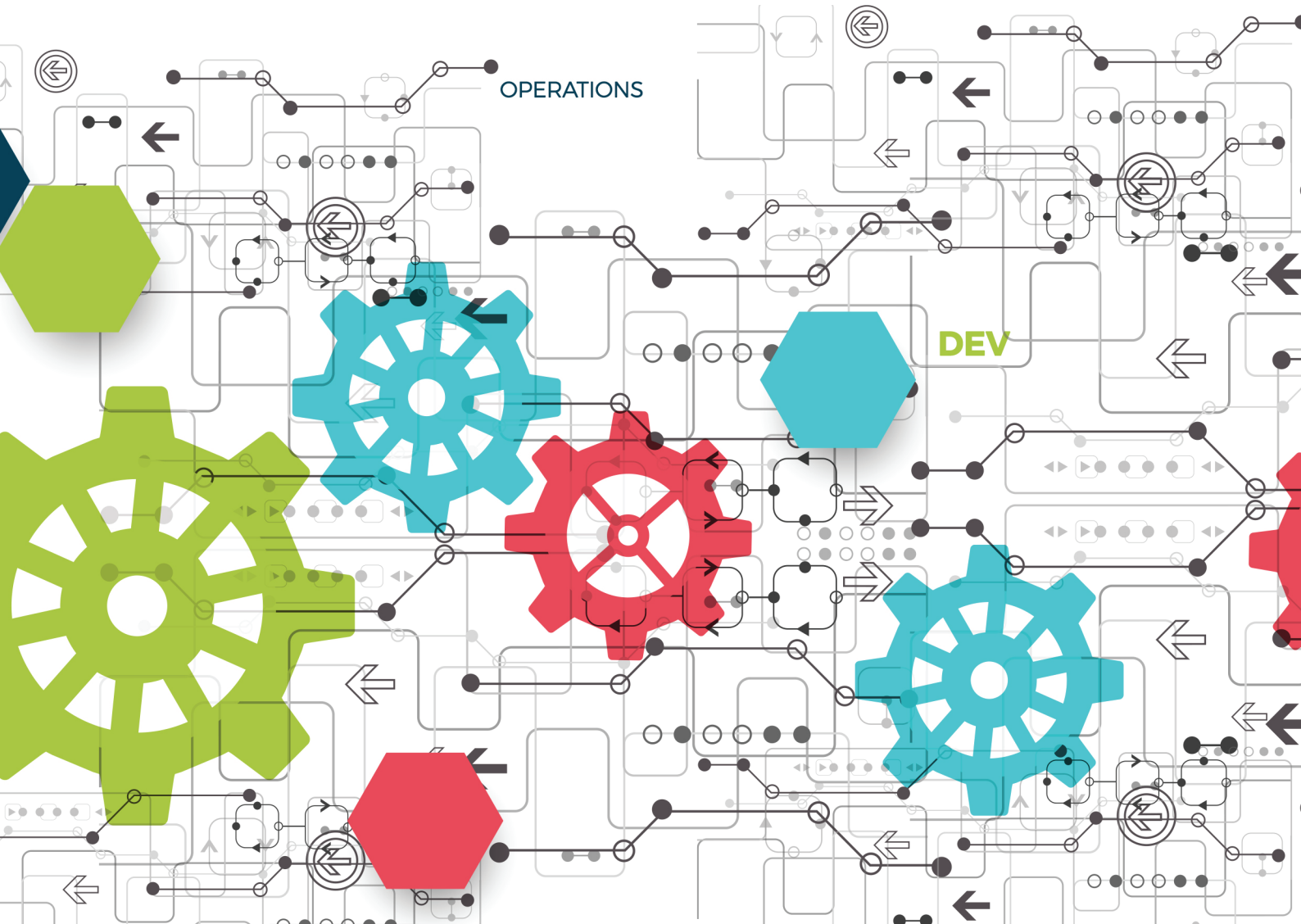
Xebia

YEAR 3

B.Tech Computer Science and Engineering in DevOps

Test Automation

MODULE 2
Introduction to Selenium



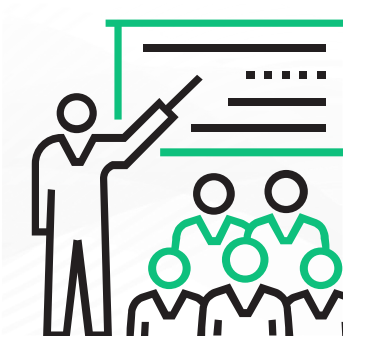
Contents

MODULE 2

Introduction to Selenium

You will learn about the ‘Software Testing’, best practices and Automation Testing

Module Objectives



At the end of this module, you will be able to learn:

- Understand the Selenium Components
- Explain Selenium Architecture
- Define TestNG and Annotations
- Understanding usage of annotations
- Describe TestSuite in TestNG
- Setting priority of execution for test cases
- Understand TestNG Reports

You will be informed about the module objectives.

At the end of this module, you will be able to learn:

- Understand the Selenium Components
- Explain Selenium Architecture
- Define TestNG and Annotations
- Understanding usage of annotations
- Describe TestSuite in TestNG
- Setting priority of execution for test cases
- Understand TestNG Reports

Module Topics

Let us take a quick look at the topics that we will cover in this module:

- Selenium Components.
- Selenium Architecture.
- What is TestNG?
- Installing TestNG in Eclipse
- TestNG Annotations
- Understanding usage of annotations
- Running a test in testing
- TestSuite in TestNG
- Setting priority of execution for test cases
- Skipping Tests
- Parameterizing Tests – DataProvider
- Putting Dataproviders for multiple tests in a single file
- Assertions/Reporting Errors
- HardAssertions
- SoftAssertions
- TestNG Reports
- What is ANT?
- Downloading and Configuring ANT
- Build.xml configuration
- XSLT Report Generation

The intention is to learn Software Automation via Selenium. Even though this presentation does its best to make you understand Selenium, but the best way is to code and learn and see how things are in real time practical labs rather than running through theories.

You will learn about the following topics in this module:

- Selenium Components.
- Selenium Architecture.
- What is TestNG?
- Installing TestNG in Eclipse
- TestNG Annotations
- Understanding usage of annotations
- Running a test in testing
- TestSuite in TestNG
- Setting priority of execution for test cases

- Skipping Tests
- Parameterizing Tests – DataProvide
- Putting Dataproviders for multiple tests in a single file
- Assertions/Reporting Errors
- HardAssertions
- SoftAssertions
- TestNG Reports
- What is ANT?
- Downloading and Configuring ANT
- Build.xml configuration
- XSLT Report Generation

1.2.1 Selenium components

These are the four components of Selenium:-

Following are the components for Selenium:

Selenium IDE – Selenium IDE(Integrated Development Environment) was a Firefox Plugin but now it even works as a plugin for Google Chrome. 'Selenese' language is used to write test scripts in IDE. In 2019, Selenium IDE is mostly referred to as a subject of learning rather than implementation in any Project. Very recently, a new version has come in the market which looks cool and it has more options where we can create multiple test suites, multiple test cases, parallel execution, better identification of locators but still the question looms will it be used for Project Level Delivery. Answer is still "NO"

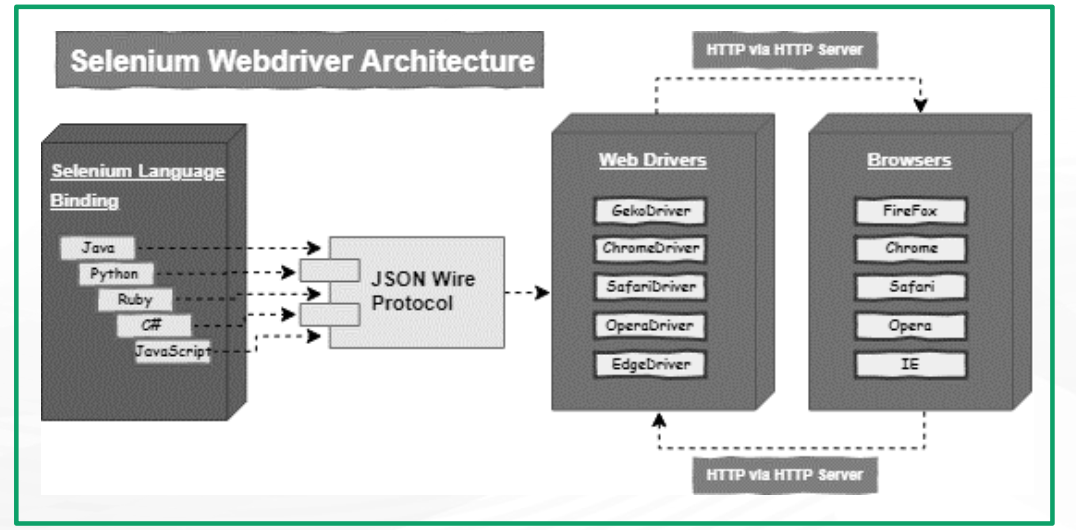
Selenium RC – Selenium Remote Control is also known as Selenium 1. It was the primary test suite used before Selenium WebDriver took its place. Even though Selenium RC stands deprecated, there can still be examples where it is used in maintenance mode. JavaScript is used for automation in Selenium. RC supports Java, JavaScript, Ruby, PHP, Python, Perl, C# and almost all the browsers.

Selenium WebDriver – The biggest change in Selenium recently has been the inclusion of the WebDriver API. WebDriver is designed in a simpler and more concise programming interface along with addressing limitations of Selenium RC. It drives the browsers much more effectively and overcomes limitations of Selenium RC for example – file upload or download scenario, handling pop-ups and alert boxes, etc.

Selenium Grid – It is part of Selenium suite that specializes in running multiple tests across different browsers, operating systems and machines in parallel. Essentially what Grid does is – “it allows running our tests in distributed test execution environment”.

1.2.2 Selenium Architecture

The major components which define Selenium are:



The major components which define Selenium are:

- **Language Bindings** - The various programming languages which are supported by Selenium e.g., Java, Python, PHP, Ruby, Perl, C#, JavaScript
- **JSON** - JavaScript Object Notation is nothing but a medium to transfer data between a Server and a Client. It is of a form of a REST API
- **BrowserDrivers** - Communicate with respective browser without revealing the source code. They receive a certain logic in the form of a command and the command is executed and it is transferred back via an HTTP response.
- **Browsers** - Selenium supports multiple browsers such as Firefox, Chrome, IE, Safari etc.,

The overall interaction among the above 4 components is what actually defines the operational procedure of Selenium.

Selenium WebDriver API helps in communication between language bindings and browsers. Each and every browser has different logic of performing actions on the browser.

Let’s understand via a small piece of code

System.setProperty(“webdriver.chrome.driver”, “path of the chromedriver.exe in your directory”);

WebDriver driver = new ChromeDriver();

driver.get(“https://www.google.co.in”);

The above code is written in Integrated Development Environment which is Eclipse. In this example Java is used.

- Once the above code is executed, every line of code will be converted as a URL. With the help of JSON Wire Protocol over HTTP. The URL's will be passed to the Browser Drivers. The client library (Java) will convert the statements of the script into JSON format and further communicate with ChromeDriver.
- Every Browser Driver uses an HTTP server to receive HTTP requests. Once the URL reaches the Browser Driver, it will then pass the request to the real browser over HTTP. Once done, the commands in Selenium script will be executed on the browser.

What did we learn so far ?

1. Selenium RC is faster than Selenium WebDriver ?
   A) True
   B) False

Answer: 1: (b)

2. Which Browser was Selenium initially started with during its launch ?
   A) Internet Explorer
   B) Firefox
   C) Google Chrome
   D) Opera

Answer: 2 : (b)

3. Which of the following is not a component of Selenium ?
   A) WebDriver
   B) RemoteControl
   C) GitHub
   D) Grid

Answer: 3 : (c)

1.2.3 What is TestNG

TestNG is a testing framework which is inspired from JUNIT and NUNIT.

Here are the new functionalities included in TestNG :

- Annotations.
- Run the tests in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class, etc...).
- Test that the code is multithread safe.
- Flexible test configuration.
- Support for data-driven testing (with @DataProvider).
- Support for parameters.
- Powerful execution model (no more TestSuite).
- Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).
- Embeds BeanShell for further flexibility.
- Default JDK functions for runtime and logging (no dependencies).
- Dependent methods for application server testing.

TestNG helps quite a lot in any Test Automation Framework to cover all categories of tests: unit, functional, end-to-end, integration. TestNG works flawlessly with all the frameworks – Data Driven, Keyword Driven, Hybrid, Cucumber, etc. It is designed to simplify a broad range of testing needs, from unit testing (testing a class in isolation of the others) to integration testing (testing entire systems made of several classes, several packages and even several external frameworks, such as application servers). TestNG is wonderful for Test Reports and the representation is rather more beautiful.

1.2.4 Installing TestNg in Eclipse

Listed below are the steps for installing TestNg:

1. Open Eclipse and go to Help
2. Help -> Install New Software.
3. Enter the URL http://beust.com/eclipse in the Work with: section and hit on Enter or click on Add Button. After some time TestNG checkbox will appear
4. Then click on Next > Next and Finish like normal routine installation
5. After sometime a window will pop up which will ask for Restart Now
6. Click on Restart Now and Eclipse will restart

Note: Expand the Right arrow of TestNG and uncheck the checkbox 'TestNG M2E (Maven) Integration (Optional)'

To determine the installation is successful:

⇥ Window > Show View > Other > Java > TestNG icon should be visible

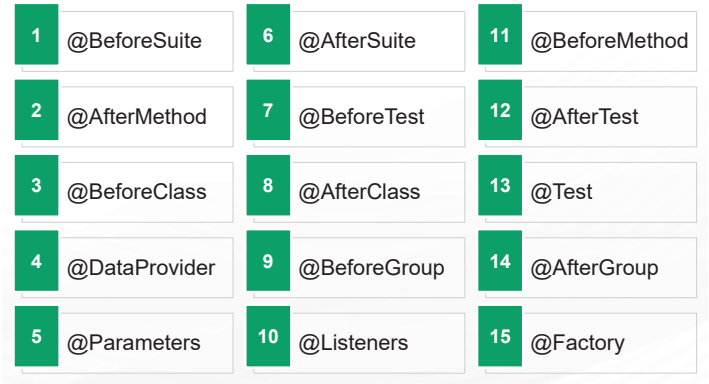
Installation of TestNG is simple. Follow the above steps it will install and work flawlessly. In mid December 2019, the url http://beust.com/eclipse was down because of some backend changes going on. In case, it goes down again in future then you can to go Eclipse Help >

Eclipse Marketplace and get TestNG too. It should work

Note : Please do not forget to uncheck 'the checkbox 'TestNG M2E (Maven) Integration (Optional)' else sometimes your tests will not run and throw an error

1.2.5 TestNG annotations

Listed below are the TestNG annotations:

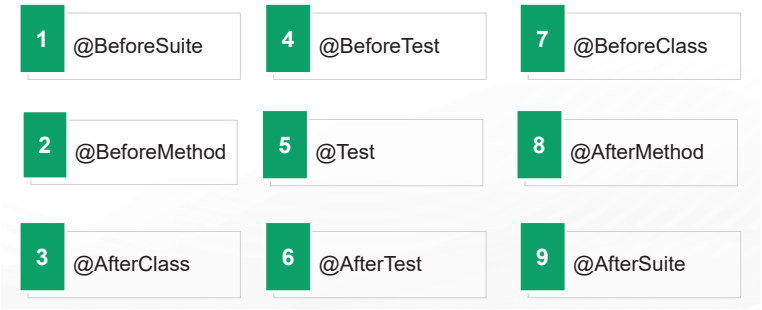


The annotations represent :

- @Test : To mark a method as a test method
- @BeforeMethod: Executes before each test method
- @AfterMethod: Executes after each test method
- @BeforeClass: Executes before the first test method in the current class
- @AfterClass: Executes after all the test methods in a current class
- @BeforeTest: Executes before any test methods of available classes
- @AfterTest: Executes after all the test methods of available classes
- @BeforeSuite: Executes before all the tests in this suite
- @AfterSuite: Executes after all tests executed in this current suite
- @BeforeGroup: Executes before the first test method run in that specified group
- @AfterGroup: Executes after the end of all the test methods executed in that specified group
- @Parameters : pass parameters to tests as arguments
- @Listeners : listens to every event that occurs in a selenium code
- @Factory : used to specify a method as a factory for providing objects to be used by TestNG for test classes. The method marked with @Factory annotation should return Object array.

1.2.6 TestNG annotations – Understanding usage

The sequence of execution of the annotations



This can be better understood when things are run in practical mode. But here if you see the above annotations, let's say a Java Class has all the above annotations mentioned then @BeforeSuite will always execute first and @AfterSuite will execute last. The sequence mentioned from 1 to 9 will run accordingly

One small thing to remember is @BeforeMethod, @Test and @AfterMethod will run as many times as there are @Test (s). So if there are 3 @Test methods then

@BeforeMethod

@Test

@AfterMethod

@BeforeMethod

@Test

@AfterMethod

@BeforeMethod

@Test

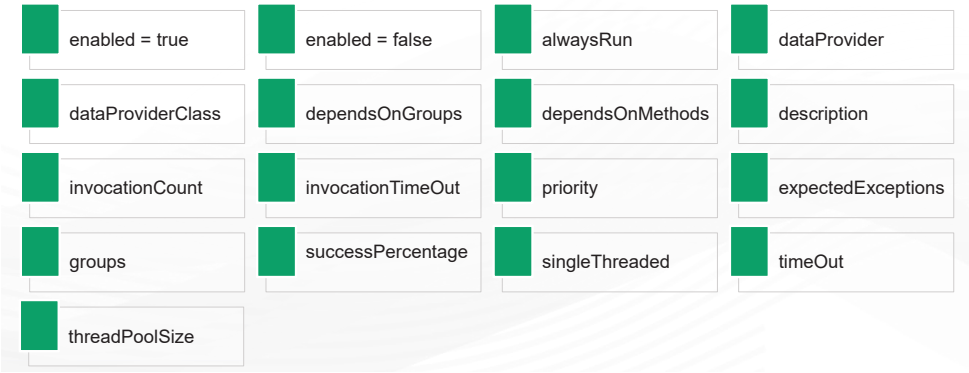
@AfterMethod

Will run 3 times for the individual @Test present in the class file.

1.2.6 TestNG annotations – Understanding usage

@Test annotation uses the maximum business logic

Attributes passed in @Test are:-



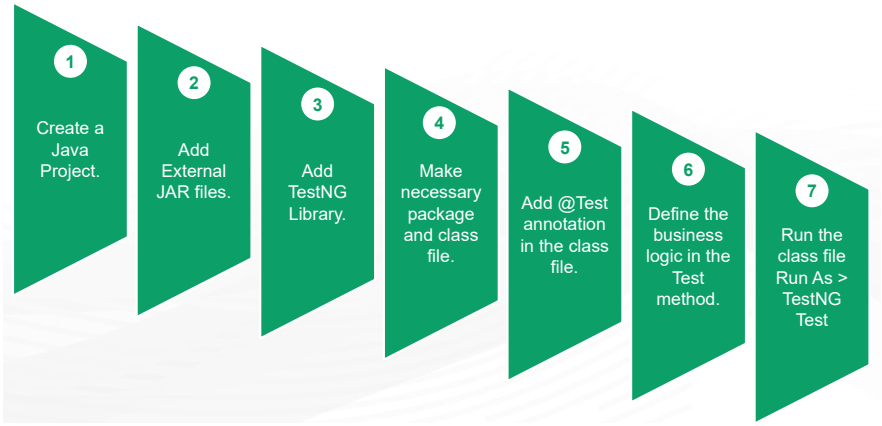
@Test annotation is the logical annotation in TestNG. Rest all annotations are supporting in terms of sequencing of execution depending on the design of the TestNG Framework. The above attributes mentioned are used but it totally depends on the tester and the nature of the logic which all attributes are to be used or even it can be skipped. It totally depends on the business login inside the @Test annotation

- **enabled = true** [this means that the @Test will run as it has been marked true enablement]
- **enabled = false** [this means that the @Test will not run as it has been marked false enablement]
- **alwaysRun** [This is used when we want to make sure a method always runs even if the parameters on which the method depends, fails. If set to true, this test method will always run]
- **dataProvider** [used for parametrization of data]
- **dataProviderClass** [This is the class from where we pass the data to data provider]
- **dependsOnGroups** [It is the list of groups this method depends on]
- **dependsOnMethods** [used to execute a method based on its dependent method]
- **description** [description for the method]
- **invocationCount** [refers to the number of times a method should be invoked kind of a loop]
- **invocationTimeOut** [refers to the maximum number of milliseconds a method should take for all the invocationCount to complete. This attribute will be ignored if invocationCount is not specified]
- **priority** [sets the priority of the test method. It is a very important attribute and is used often in cases of multiple Tests]
- **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]
- **successPercentage** [The percentage of success expected from this method]

- **singleThreaded** [If set to true, all the methods on this test class are guaranteed to run in the same thread, even if the tests are currently being run with parallel=”methods”. This attribute can only be used at the class level and it will be ignored if used at the method level. **Note: this attribute used to be called sequential (now deprecated)]**
- **timeOut** [The maximum number of milliseconds this test should take]
- **threadPoolSize** [The size of the thread pool for this method. The method will be invoked from multiple threads as specified by invocationCount. **Note: this attribute is ignored if invocationCount is not specified]**

1.2.7 Running a Test in TestNg – JAVA Project

Running a TestNG Test for a basic Java Project



The prerequisites needed to run a test in TestNG

Let’s say for example we are using a simple Java Project. And we make a package and a class file. Now, we want to run the class file via TestNG. So what we need to do is add the TestNG jar file in the project and also add TestNG Library.

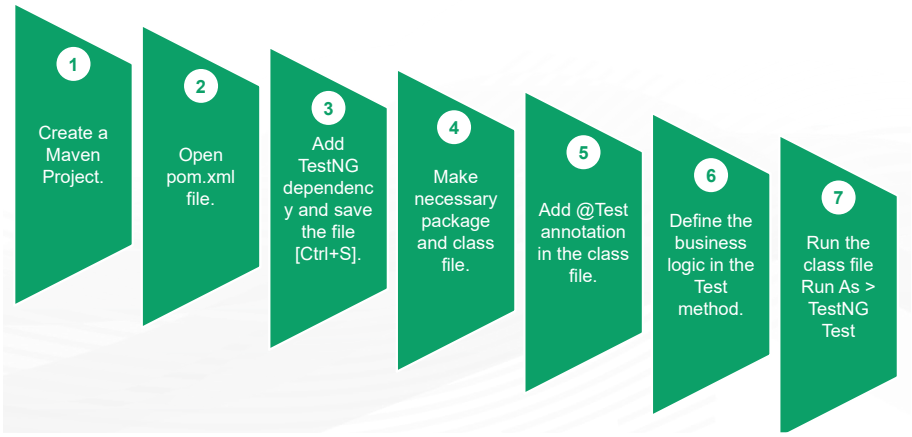
Give just a @Test annotation and apply the business logic inside it.

Add the information about the test which is essential. Then right click on the class file and Run as TestNg Test.

See the output console.

1.2.7 Running a Test in TestNg – MAVEN Project

Running a TestNG Test for a Maven Project



The basic difference here is that there is no need to add an external jar file for TestNG in this project, instead can add TestNG Maven dependency in the pom.xml file and save the file. By default, all the TestNG Libraries will be imported in the project.

1.2.7 Running a Test in TestNg – create testng.xml

To create testng.xml file

- Right click on your Project and New > File > testng.xml
- Right click on your Project > TestNG > Convert to TestNG

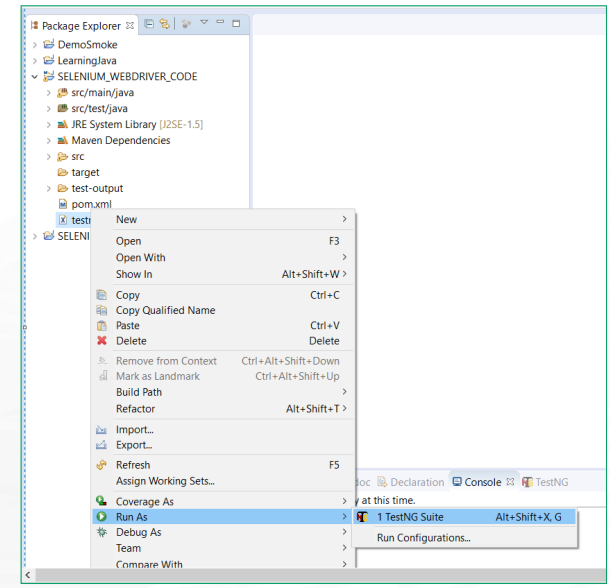
```
<suite name = "name of test suite">
<test name = "name of the test">
<classes>
<class name = "packagename. classname"/>
</classes>
</test>
</suite>
```

Testng.xml file is a very sensitive file which allows to configure a test suite and execute it from the IDE or command line or ANT script

When a TestNG test is executed through Eclipse or through Maven build, HTML reports are generated. These HTML reports also create a test suite XML file that is used to execute the same test through the command line

1.2.7 Running a Test in TestNg – Execute testng.xml

Right click on testng.xml file > Run As > TestNG Suite



As soon as we do Run As > TestNG Suite, testNG execution engine starts and soon we will notice that our test will run and complete.

Once the execution is complete, we can view test execution results under the TestNg console.

1.2.8 TestSuite in TestNG

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite thread-count="1" verbose="1" name="AutomationPractice Suite" annotations="JDK" parallel="tests">
<test name="RegisterUser">
            <classes>
              <class name="com.suite1.Registration"/>
            </classes>
  </test>
<test name="PaymentConfirmation">
    <classes>
      <class name="com.suite2.Payment"/>
    </classes>
  </test>
</suite>
```

Follow the steps below:

1. Make a JAVA Project
2. Make two packages inside it and name them.
3. Make few classes inside them and name them accordingly.
4. Make your testng.xml file and follow the above testng.xml file which is in the slide and design yours accordingly. Careful with the syntax as .xml is a sensitive file.
5. Run the testng.xml file

Let's understand the terms mentioned in the .xml file.

1. **thread-count:** This is used for parallel execution, based on the number scripts. It will execute in parallel or sequential order. Try making it 2 or 3 and then you see for yourself.
2. **verbose:** It is used to log the execution details in the console. The value should be 1-10. The log details in the console window will get more detailed and clearer as you increase the value of the verbose attribute in the testng.xml configuration file.
3. **name**: Name of the suite. Here it is "AutomationPractice Suite"
4. **Parallel**: To run scripts parallel, value can be tests/classes/methods/suites.

Here is another simple example for better understanding:

<suite name = "AutomationPractice">

<test name = "Module1Test">

<classes>

<class name = "packagename. Login"/>

<class name = "packagename. Home"/>

<class name = "packagename. MenClothing"/>

<class name = "packagename. WomenClothing"/>

</classes>

</test>

</suite>

1.2.9 Setting priority of execution for test cases

```
@Test(priority = 1)
public void Login() {
   //driver.findElement(By.xpath("//nav[@id = 'mainNav']/descendant::ul/child::li[4]/child::a[1]")).click();
   driver.findElement(By.xpath("//input[@id = 'UserName']")).sendKeys("admin");
   driver.findElement(By.xpath("//input[@id = 'Password']")).sendKeys("password");
   driver.findElement(By.xpath("//div[@id = 'loginContainer']/descendant::div[20]/a[1]")).click();
   System.out.println(driver.getTitle());
   String GetTitle = "WiZDOM 7.0";
   softassert.assertEquals(driver.getTitle(), GetTitle, "Title is Wrong");
   softassert.assertAll();
}


@Test(priority = 2)
public void ManageUsersPage() throws Exception {
   Thread.sleep(2000);
   driver.findElement(By.xpath("//div[@id = 'sidebar']/descendant::a[2]")).click();
   driver.findElement(By.xpath("//div[@id = 'sidebar']/descendant::a[3]")).click();
   softassert.assertEquals(driver.findElement(By.xpath("//input[@id = 'UserMode']/following::span[1]")).getText(), "Manage Users");
   softassert.assertAll();
}

@Test(priority = 3)
public void AddUser() throws Exception {
   driver.findElement(By.xpath("//span[@id = 'headerPlaceHolder']/parent::div/descendant::li[2]/a")).click();
   Select select = new Select(driver.findElement(By.xpath("//select[@id = 'UserTitle']")));
   select.selectByVisibleText("Doctor");
   driver.findElement(By.xpath("//input[@id = 'FirstName']")).sendKeys("Sarthak");
   driver.findElement(By.xpath("//input[@id = 'MiddleName']")).sendKeys("Kumar");
   driver.findElement(By.xpath("//input[@id = 'LastName']")).sendKeys("Panda");
```

Here three test annotations are created and the method names associated with them are Login(), ManageUsersPage() and AddUser().

When testng.xml file is run, then which one will run when, is purely determined by randomness in case there is no prioritization of the execution order. Sometimes testng runs these tests in alphabetical order but the case is not 100% true. Our objective is to run them in a sequential manner and in a manner which we want to run.

So let's say we want Login() to run first

ManageUsersPage() to run second

AddUser() to run third

So we need to do a small workaround to achieve this and it is quite simple. Just against the @Test annotation you need to insert (priority = n) where 'n' is the priority value.

Once you do this and run the tests then as per the priority set, the test cases shall run likewise

1.2.10 Skipping Tests

```
@Test(enabled = true)
public void Login() {
    //driver.findElement(By.xpath("//nav[@id = 'mainNav']/descendant::ul/child::li[4]/child::a[1]")).click();
    driver.findElement(By.xpath("//input[@id = 'UserName']")).sendKeys("admin");
    driver.findElement(By.xpath("//input[@id = 'Password']")).sendKeys("password");
    driver.findElement(By.xpath("//div[@id = 'loginContainer']/descendant::div[20]/a[1]")).click();
    System.out.println(driver.getTitle());
    String GetTitle = "WiZDOM 7.0";
    softassert.assertEquals(driver.getTitle(), GetTitle, "Title is Wrong");
    softassert.assertAll();
}

@Test(enabled = false)
public void ManageUsersPage() throws Exception {
    Thread.sleep(2000);
    driver.findElement(By.xpath("//div[@id = 'sidebar']/descendant::a[2]")).click();
    driver.findElement(By.xpath("//div[@id = 'sidebar']/descendant::a[3]")).click();
    softassert.assertEquals(driver.findElement(By.xpath("//input[@id = 'UserMode']/following::span[1]")).getText(), "Manage Users");
    softassert.assertAll();
}

@Test(enabled = true)
public void AddUser() throws Exception {
    driver.findElement(By.xpath("//span[@id = 'headerPlaceHolder']/parent::div/descendant::li[2]/a")).click();
    Select select = new Select(driver.findElement(By.xpath("//select[@id = 'UserTitle']")));
    select.selectByVisibleText("Doctor");
    driver.findElement(By.xpath("//input[@id = 'FirstName']")).sendKeys("Sarthak");
    driver.findElement(By.xpath("//input[@id = 'MiddleName']")).sendKeys("Kumar");
    driver.findElement(By.xpath("//input[@id = 'LastName']")).sendKeys("Panda");
```

There are three test annotations created.The method names associated with them are Login(), ManageUsersPage() and AddUser().

Let's say you want to skip ManageUsersPage() then you have to write (enabled = false) against the @Test annotation for that test.

So this particular test will be skipped

In the previous slide, there was priority and now in this slide the priority has been removed and enabled = true and enabled = false has been introduced. So what happens to priority now ?

Answer is @Test(priority = 1, enabled = true)

@Test(priority = 2, enabled = false)

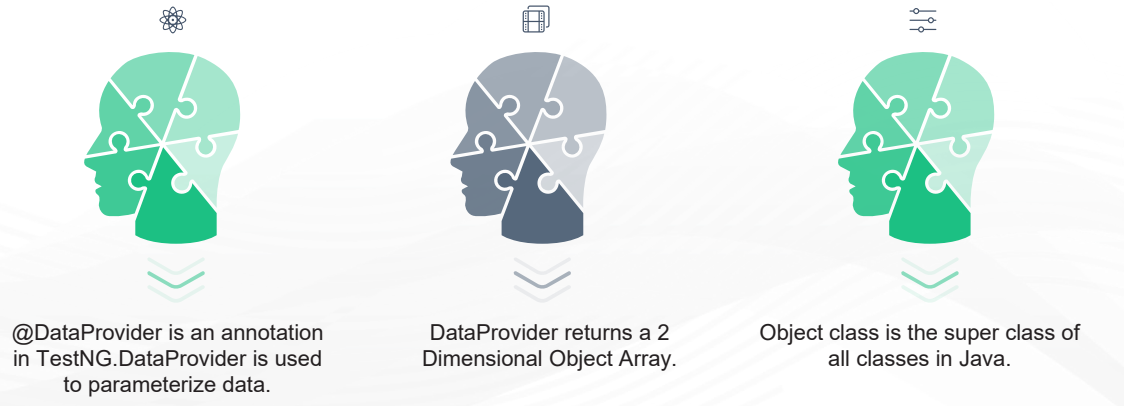
@Test(priority = 3, enabled = true)

So it will work. Try it and run the testng.xml file and see for yourself. Also if a question is arising in your head that the 2nd priority is getting skipped so what will happen with the priority = 3 –> will it take a precedence, will something new come in the output console.

The best way to answer this question is to see for yourself by coding the same.

1.2.11 Parameterizing Tests - DataProvider

Parameterization is executing one test with multiple sets of data



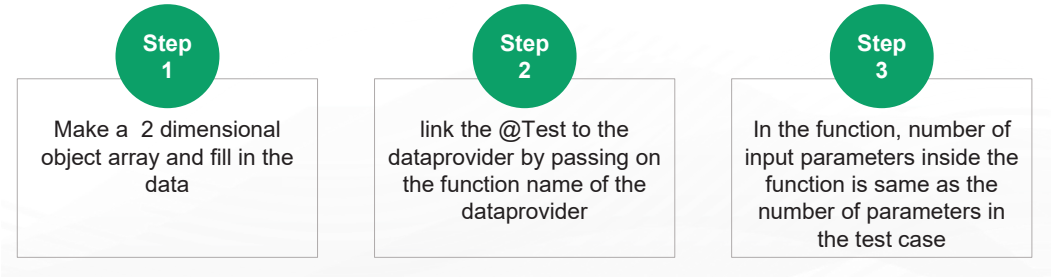
DataProvider is used to parameterize data. Parameterization means executing one test with multiple sets of data. Let's say in an excel file there are multiple sets of data and you want to run all those multiple sets of data for a single test case which you have mentioned in your code. This is known as parameterization

@DataProvider is imported from the package **import org.testng.annotations.DataProvider;**

@DataProvider returns a 2 dimensional Object Array

1.2.11 Parameterizing Tests - DataProvider

Designing Data Provider needs 3 basic steps:-



Here is a sample code:

package browsers;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

public class LearningDataProvider {

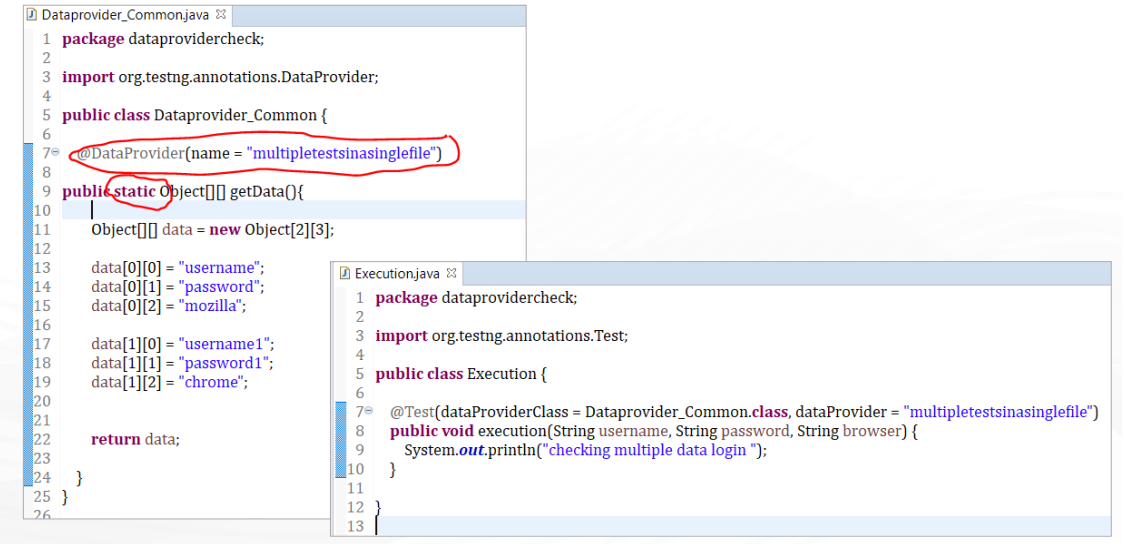
@Test(dataProvider = "getData")

public void Login(String username, String password, String browser, String result) {

}


```
@DataProvider
public Object[][] getData(){
Object[][] obj = new Object[3][4];
obj[0][0] = “username”;
obj[0][1] = “password”;
obj[0][2] = “mozilla”;
obj[0][3] = true;
obj[1][0] = “username1”;
obj[1][1] = “password1”;
obj[1][2] = “chrome”;
obj[1][3] = true;
obj[2][0] = “username2”;
obj[2][1] = “password2”;
obj[2][2] = “microsoft edge”;
obj[2][3] = true;
return obj;
}
}
```


1.2.12 – Putting Data Providers for multiple tests in a single file



Dataprovider_Common.java

```
package dataprovidercheck;

import org.testng.annotations.DataProvider;

public class Dataprovider_Common {

  @DataProvider(name = "multipletestsinasinglefile")

public static Object[][] getData(){
    Object[][] data = new Object[2][3];

    data[0][0] = "username";
    data[0][1] = "password";
    data[0][2] = "mozilla";

    data[1][0] = "username1";
    data[1][1] = "password1";
    data[1][2] = "chrome";


    return data;

  }
}
```

Execution.java

```
package dataprovidercheck;

import org.testng.annotations.Test;

public class Execution {

  @Test(dataProviderClass = Dataprovider_Common.class, dataProvider = "multipletestsinasinglefile")
  public void execution(String username, String password, String browser) {
    System.out.println("checking multiple data login ");
  }

}
```

Let's say for example you are working on a project in which you have created 100 classes having 100 tests which all need lots of data to be run. Is it a viable option to write @DataProvider in all the tests ? Answer can be yes but it is not a fast option. So there is a certain workaround which can be done to get this thing sorted. Let's look at same practically. Please follow the steps :

1. Make a Java Project and make a package and make 2 classes
2. Name one class as Execution
3. Name the other class as DataProvider_Common
4. Create an @Test annotation in the Execution class
5. Create an @DataProvider annotation in the DataProvider_Common class
6. Create a 2 dimensional Object array
7. The only change which you need to make is make it static so that it can be called directly from class name [refer to the image above]
8. Also give a valid name to the dataprovider. I have given here @DataProvider(name = "multipletestsinasinglefile")
9. Now go to the Execution class and in the @Test you have to give dataProviderClass name and the name of the data provider from which the data has to be derived[please refer to the image attached]
10. @Test(dataProviderClass = Dataprovider_Common**.class, dataProvider = "multipletestsinasinglefile")**
11. Now execute the class Execution. See the output. It will show something like below

    [RemoteTestNG] detected TestNG version 6.14.3

    checking multiple data login

    checking multiple data login

PASSED: execution(“username”, “password”, “mozilla”)

PASSED: execution(“username1”, “password1”, “chrome”)

==========

Default test

Tests run: 2, Failures: 0, Skips: 0

==========

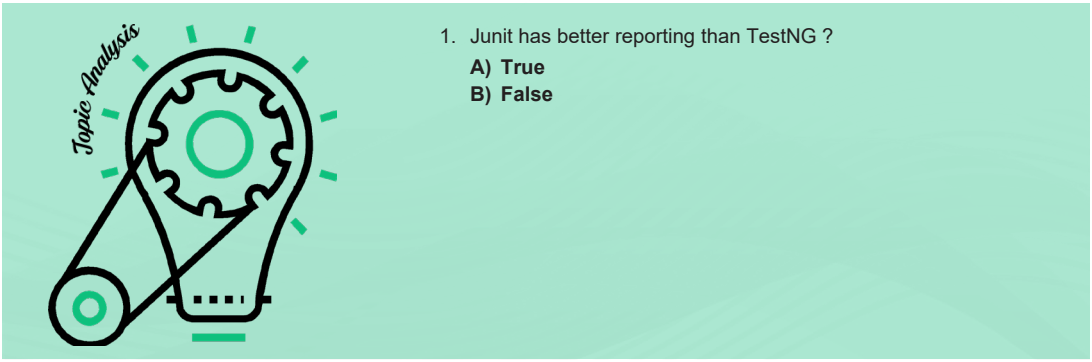
- ==========
- Default suite
- Total tests run: 2, Failures: 0, Skips: 0
- ==========

1.2.13 Running a Test in TestNg – Different Ways

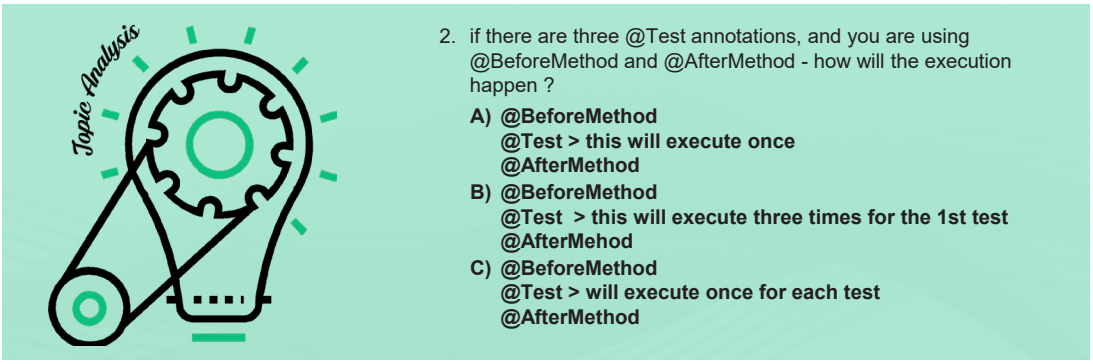
TestNG can be invoked via

Automation Testers use Eclipse IDE to do most of their testing operations. ANT and Command Line are sparingly used. So we will stick to run testng.xml file and it almost covers single, multiple tests and that too we can run in parallel.

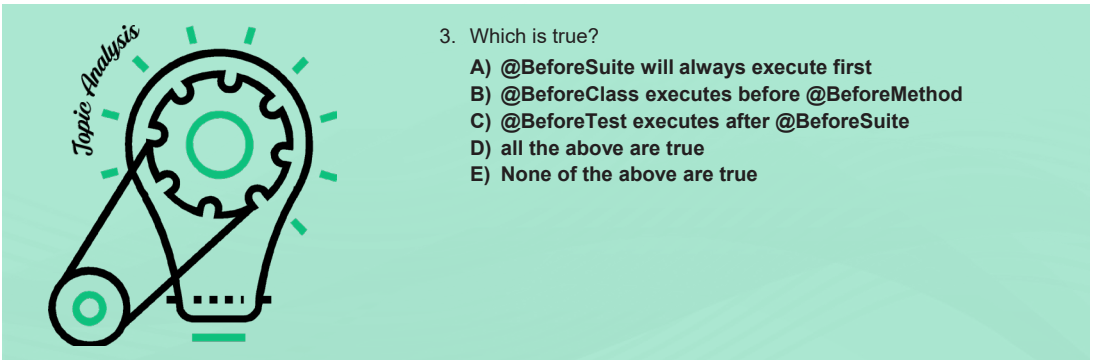
What did you learn so far?



Answer: 1 : (b)



Answer: 2 : (c)



Answer: 3 : (d)

1.2.14 HardAssertion

HardAssertion terminates execution the moment a condition has not been met

- HardAssertion does not allow execution of the other test cases irrespective they are logically correct or not.
- The test case will be immediately marked as Failed when a hard assertion condition fails.
- It is not a flexible option to use in Selenium Automation Scripts.
- Assert is the keyword used for HardAssertion.

Assert assert = new Assert(); is the correct way to write. Hard Assert does not mean that the line of code be like this HardAssert assert = new HardAssert();

Thing with such hard assertion is let's say for example there are 20 test cases and your 5th test case fails. Hard assert will not test from 6 to 20th. It will just come out and give the results upto 5th test case and skip the rest of it.

1.2.15 SoftAssertion

A soft assertion continues with test execution even if the assertion condition is not met

- Soft assertion does not terminate anywhere in between, it continues till all the test cases have been executed.
- Soft assertion is quite flexible.
- The object of SoftAssert class has to be created.
- Every test case has to have the method assertAll().

Soft Assertion is more practical to use as there could be thousands of test cases and all test cases should not be terminated. In case, in the beginning one test cases fails First create an object of the SoftAssert class. It can be declared globally or at a local level.

SoftAssert softassert = new SoftAssert();

Then using the object reference 'softassert' can call all the assertion methods for example:-

- assertEquals
- assertTrue
- assertFalse
- assertNull
- assertNotNull
- assertSame
- assertNotSame
- assertArrayEquals

But something to note which is important is for every test case when we use Soft Assertion we have to use the following line of code

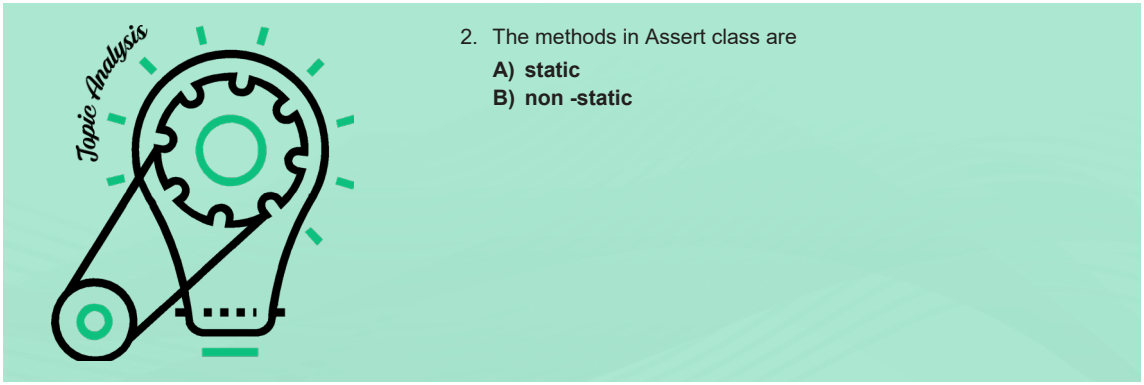
softassert.assertAll();

This will collate all the assertion failures for the same object at a single time

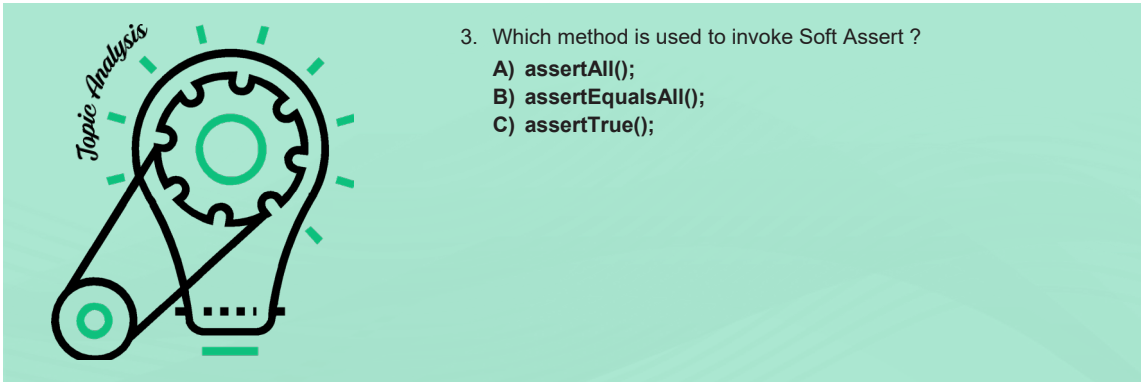
What did you learn so far ?

1. We need to make object of Assert to access it ?
   A) true
   B) false

Answer: 1: (a)

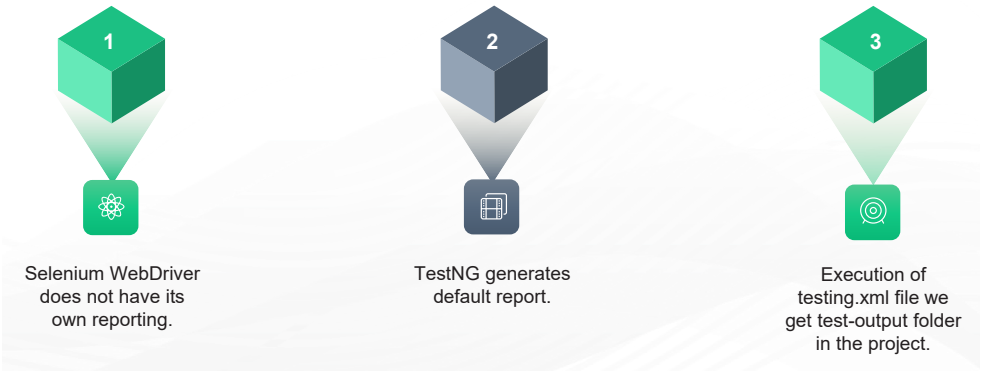


Answer: 2: (a)



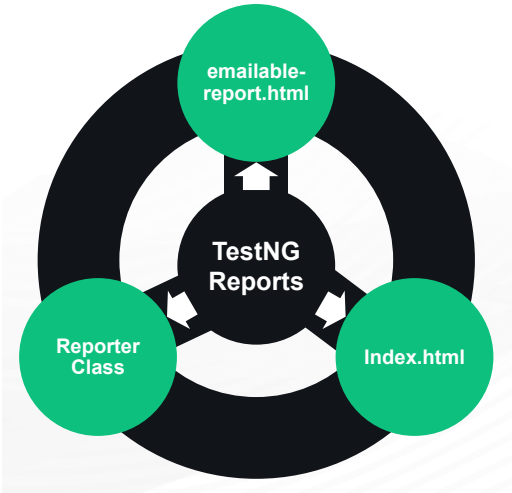
Answer: 3: (a)

1.2.16 TestNG Reports



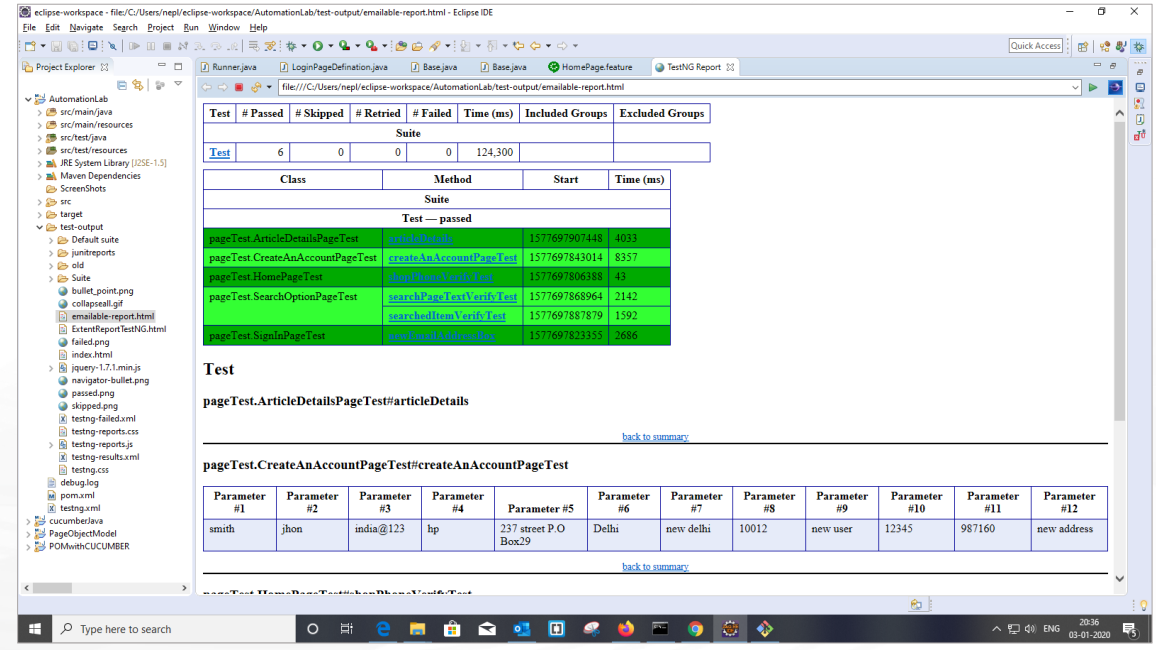
- Generation of reports is very important while doing Automation Testing as no end client will be interested in seeing the code rather they will see the reports.
- Reports give complete statistics of how many test cases - passed, failed and skipped.
- Report gives the status of the project.
- The TestNG will generate the default report.
- When executing testng.xml file, and refresh the project. A test-output folder will be there in that folder.

1.2.16 TestNG Reports - types

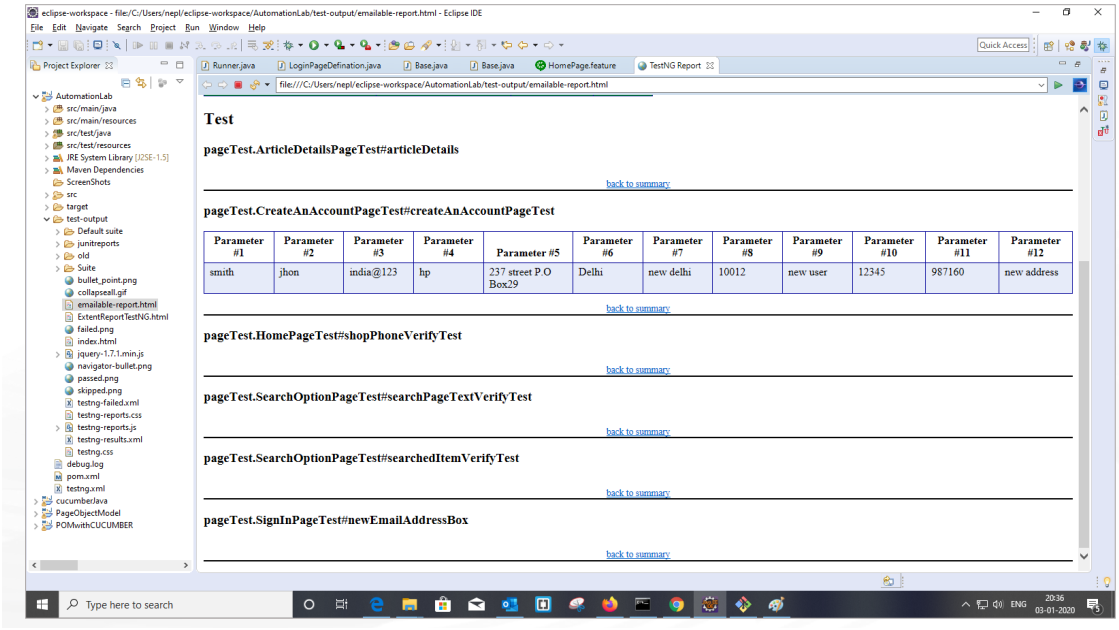


These reports give statistics of the execution with graphical and pictorial representation.

1.2.16 TestNG Reports - emailable-report.html

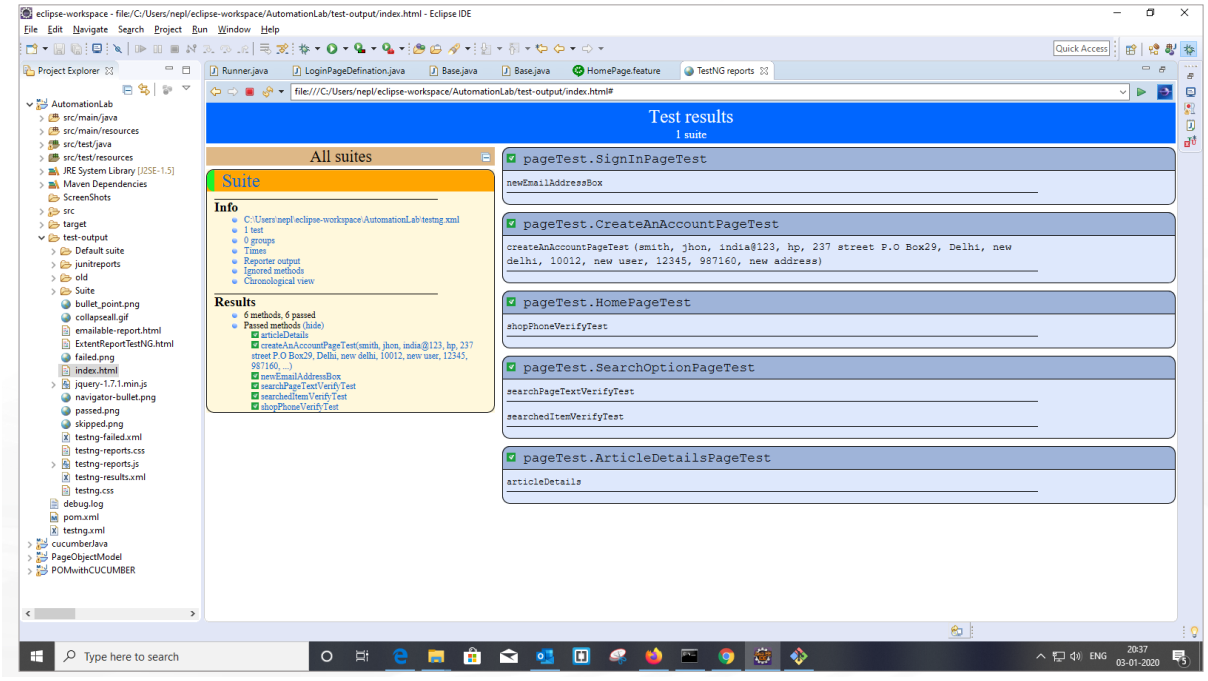


- Click on option “emailable-report.html”
- Click on option web browser



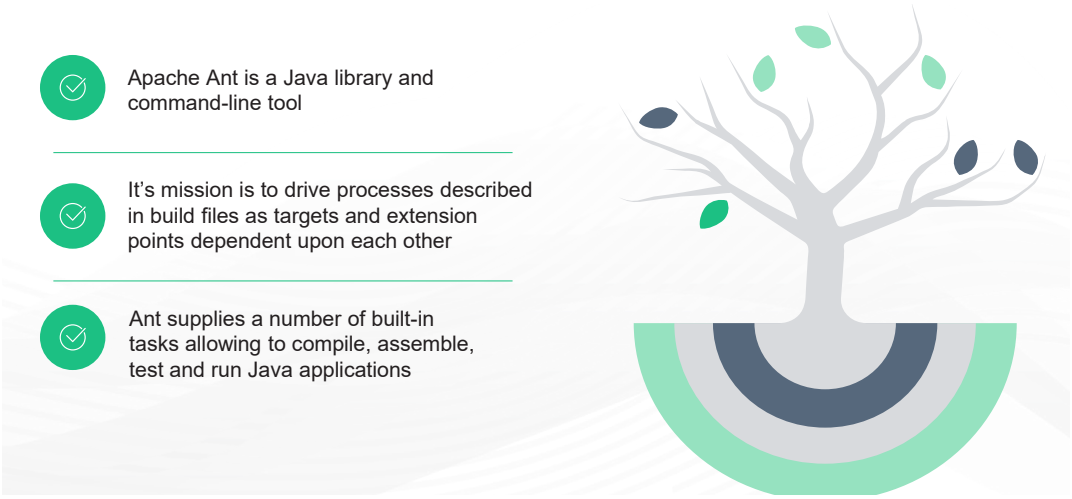
- Click on option “emailable-report.html”
- Click on option web browser

1.2.16 TestNG Reports - index.html



- Right click on the index.html from the project directory.
- Select option open with web browser option. It will display the result in a specific order.
- Tester can check as per his requirement.

1.2.18 What is ANT?

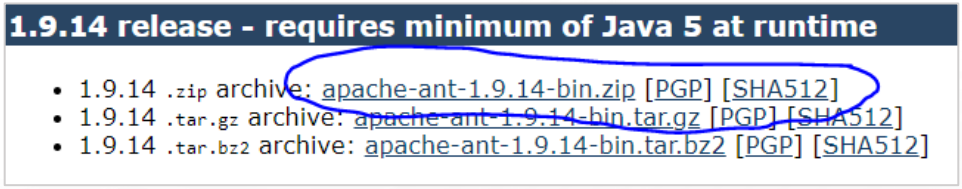


Here are the key points about Apache Ant:

- Apache Ant is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of Ant is the build of Java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications. Ant can also be used effectively to build non Java applications, for instance C or C++ applications. More generally, Ant can be used to pilot any type of process which can be described in terms of targets and tasks.
- Ant is written in Java. Users of Ant can develop their own "antlibs" containing Ant tasks and types, and are offered a large number of ready-made commercial or open-source "antlibs".
- Ant is extremely flexible and does not impose coding conventions or directory layouts to the Java projects which adopt it as a build tool.
- Software development projects looking for a solution combining build tool and dependency management can use Ant in combination with Apache Ivy
- The Apache Ant project is part of the Apache Software Foundation

1.2.19 Downloading ANT

- Download ANT from google
- Download the latest Zip file and extract it



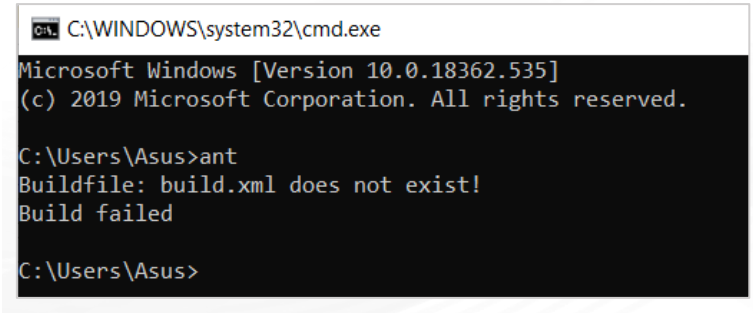
Open Google in any browser and type 'Download Apache ANT'.

Go to https://ant.apache.org/bindownload.cgi and select a .zip archive (the latest one) e.g., 1.9.14 .zip archive: apache-ant-1.9.14-bin.zip [PGP] [SHA512].

Wait as it will get downloaded into your System.

1.2.19 Configuring ANT

- Copy the directory path of the ANT Folder.
- Setup the Environment Variables.
- Check in command prompt for installation



Extract the zip file which was downloaded and save it anywhere in your System

In the Environment Variables:-

ANT_HOME

give the path - D:\SELENIUM\APACHE ANT\apache-ant-1.9.14

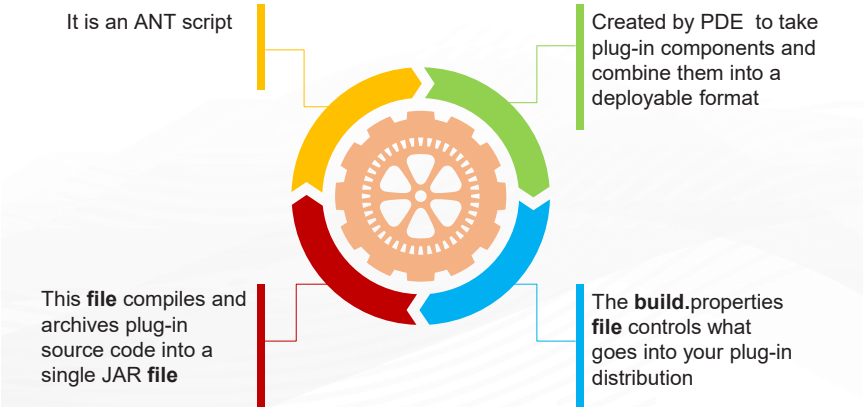
then edit the path variable and give the bin path

D:\SELENIUM\APACHE ANT\apache-ant-1.9.14\bin

Open a command prompt and type ant and if you see build.xml file does not exist! then it is a successful installation

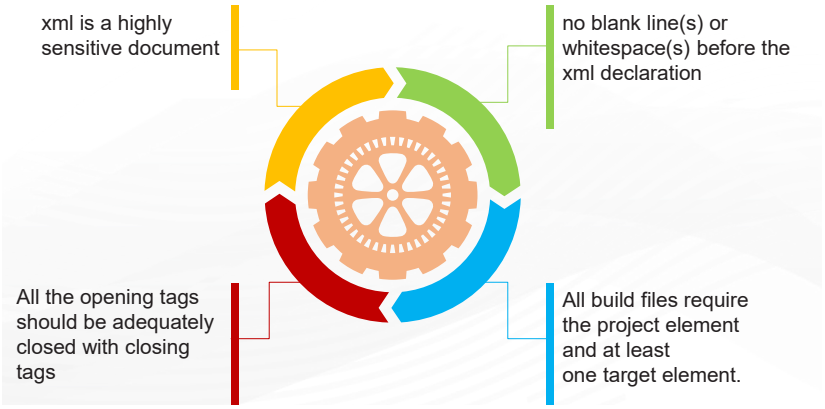
1.2.19 Build.xml configuration

Build.xml is the heart of ANT. Some Key points about Build.xml




```
<?xml version = “1.0”?>
 <project name = “Learning Selenium Project” default = “info”>
<target name = “info”>
<echo>Learning ANT in Selenium</echo>
</target>
</project>
```

Points to be remembered while configuring the Build.xml



Typically, Ant’s build file, called build.xml should reside in the base directory of the project. However there is no restriction on the file name or its location. You are free to use other file names or save the build file in some other location.

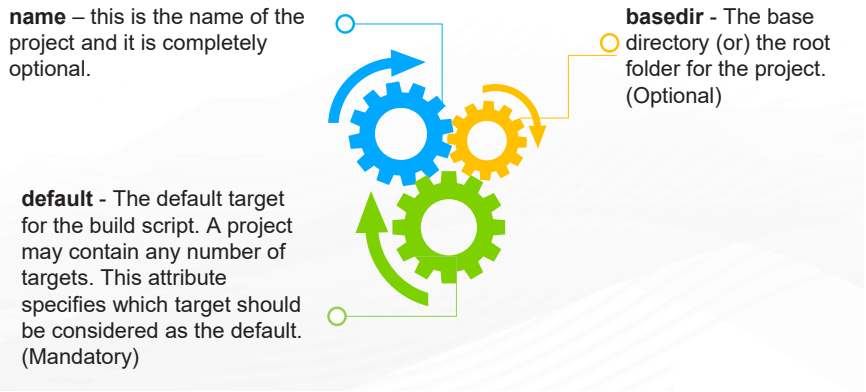
Typically there should be no blank line(s) or whitespace(s) before the xml declaration. In case there is , the following error message occurs while executing the ANT build -

“The processing instruction target matching “[xX][mM][lL]” is not allowed.”

- A target is a collection of tasks that you want to run as one unit. In our example, we have a simple target to provide an informational message to the user.
- Targets can have dependencies on other targets. For example, a **deploy** target may have a dependency on the **package** target, the **package** target may have a dependency on the compile target and so forth. Dependencies are denoted using the **depends** attribute

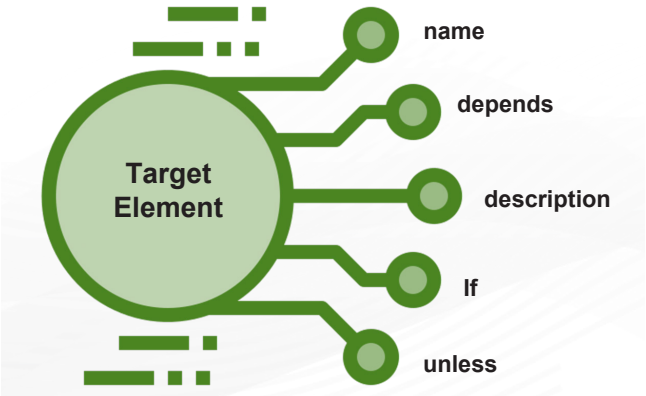
1.2.19 Build.xml configuration – Project element

Xml element project has three attributes



1.2.19 Build.xml configuration – Target element

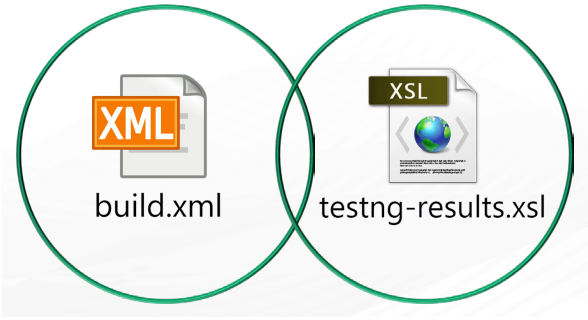
The target element has the following attributes:



The **target** element has the following attributes:

- **name** - The name of the target (required).
- **Depends** - Comma separated list of all targets that this target depends on (optional).
- **Description** - A short description of the target (optional).
- **if** - Allows the execution of a target based on the trueness of a conditional attribute (optional).
- **Unless** - Adds the target to the dependency list of the specified Extension Point. An Extension Point is similar to a target, but it does not have any tasks (optional).

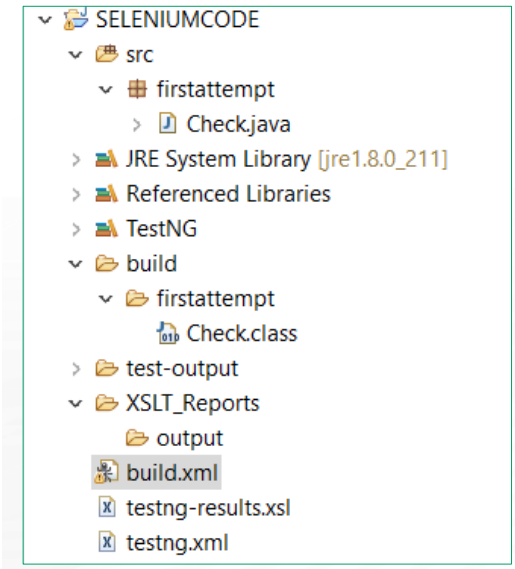
1.2.21 XSLT report generation generation using TestNg and Ant



Attached **build.xml** and **testng-results.xsl** files.

Let's learn something about how to go about creating XSLT reports and also a little bit about build.xml file and testng-results.xsl file in practical situations

Adding build.xml file to project



First copy the build.xml file in your project. You can see the image that is added in build.xml file in the project 'SELENIUM CODE'

Changes that needs to be done in build.xml file

build.xml

```
1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <!DOCTYPE project [
3  ]>
4
5  <project name="SELENIUM CODE" default="usage" basedir=".">
6
7  <!-- ========== Initialize Properties ==================================== -->
8    <property environment="env"/>
9
10   <property name="ws.home" value="${basedir}"/>
11   <property name="ws.jars" value="D:\SELENIUM\Selenium JAR Files"/>
12   <property name="test.dest" value="${ws.home}/build"/>
13   <property name="test.src" value="${ws.home}/src"/>
14   <property name="ng.result" value="test-output"/>
15
```

Follow these steps:

- Open the build.xml file which you have added in the project.
- As highlighted in the image change the project name
- As highlighted in the image give the path where you have kept your JAR Files.

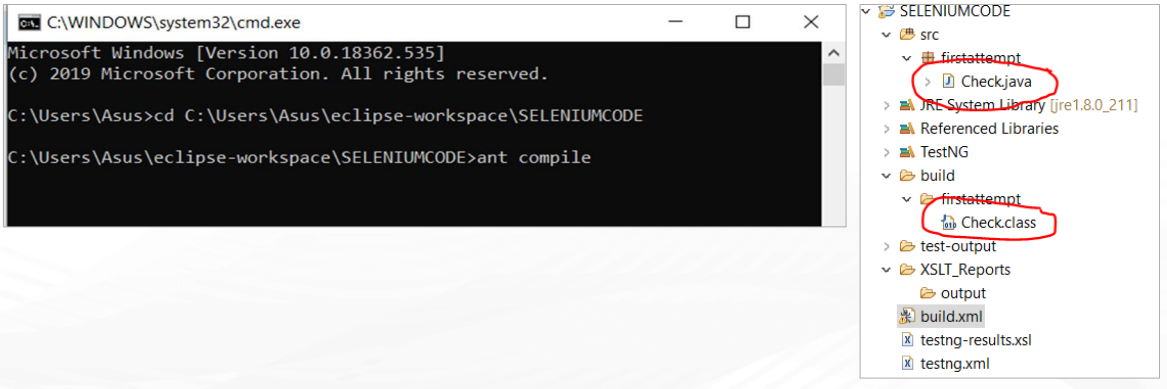
Target name – “run” in build.xml file

```
<!-- run -->
<target name="run" depends="compile">
  <testng classpath="${test.classpath}:${test.dest}" suitename="suite1">
    <xmlfileset dir="${ws.home}" includes="testng.xml"/>
  </testng>
  <!--
```

If you scroll down the build.xml file, you can see a target known as run.

This target runs as per testng.xml

Alongside include this statement <xmlfileset dir=”${ws.home}” includes=”testng.xml”/> , give the path of the testng.xml file.



Always open a fresh command prompt while working with ANT

cd pathof the project

ant compile

Once you see BUILD success, refresh the project build you will see a build folder in the project. It will have class files corresponding to all the java files

compile target converts the java files to class files

- First it deletes the test.destination directory.
- It makes the build directory again.
- Then it compiles the complete project.
- It converts all java files to class files.

See the image clearly.Here we have mentioned one java file and post compiling there is a class file in the build folder. In case, you have multiple java files, then you can see multiple class files in the build folder.

- XSLT reports are better reports than testng reports
- Need to copy testng-results.xsl file in the project

```
<target name="makexsltreports">
   <mkdir dir="${ws.home}/XSLT_Reports/output"/>

   <xslt in="${ng.result}/testng-results.xml" style="testng-results.xsl"
      out="${ws.home}/XSLT_Reports/output/index.html" classpathref="test.c" processor="SaxonLiaison">
     <param name="testNgXslt.outputDir" expression="${ws.home}/XSLT_Reports/output/"/>
     <param name="testNgXslt.showRuntimeTotals" expression="true"/>
   </xslt>
 </target>
```

There is Nno need to do any changes in the testng-results.xsl file at all.Make sure that in the build.xml file under the target name = “makexsltreports”, give the path of the testng-results. xsl file in style as shown in the image.

To generate xslt reports use the following command in cmd

- ant makexsltreports

```
C:\Users\Asus\eclipse-workspace\SELENIUMCODE>ant makexsltreports
```

After adding the testng-results.xsl file in the project and doing all the necessary changes in the build.xml file,

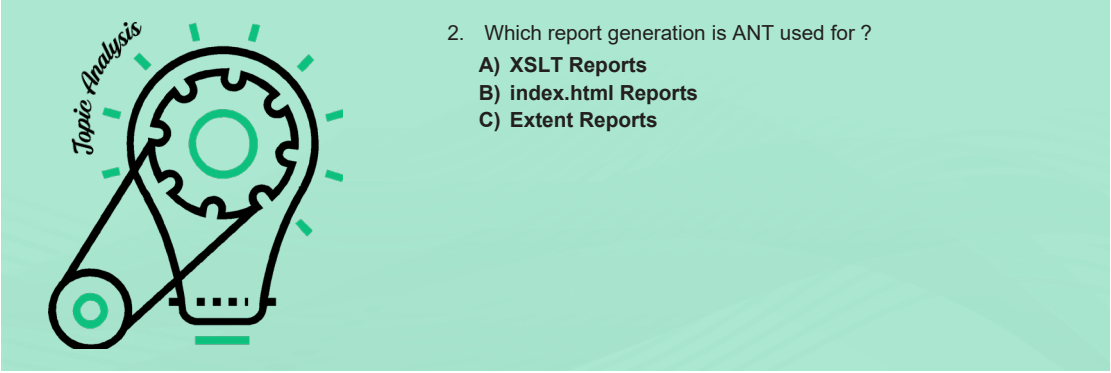
Go to cmd and type the command : ant makexsltreports and press enter

XSLT_Reports folder will get generated in the project along with an output folder and it will have the xslt report which is quite eye catching

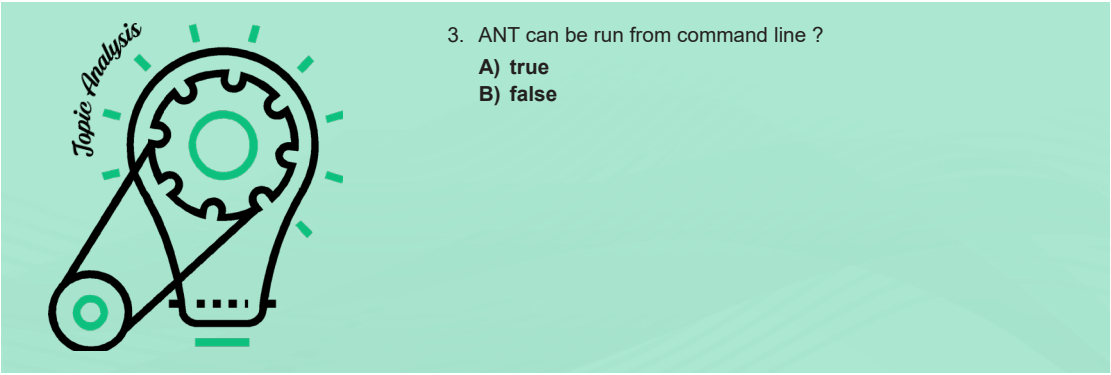
What did you grasp ?

1. Heart of ANT ?
   - **A) build.xml**
   - **B) pom.xml**
   - **C) testng.xml**

Answer: 1: (a)



Answer: 2 : (a)



Answer: 3 : (a)

In a nutshell, we learnt:

1. Selenium Components.
2. Selenium Architecture.
3. What is TestNG?
4. Installing TestNG in Eclipse
5. TestNG Annotations
6. Understanding usage of annotations
7. Running a test in testing
8. TestSuite in TestNG
9. Setting priority of execution for test cases
10. Skipping Tests
11. Parameterizing Tests – DataProvide
12. Putting Dataproviders for multiple tests in a single file
13. Assertions/Reporting Errors

14. HardAssertions
15. SoftAssertions
16. TestNG Reports
17. What is ANT?
18. Downloading and Configuring ANT
19. Build.xml configuration
20. XSLT Report Generation

Now, you have reached the end of the module, In this module, you have learned:

- Selenium Components.
- Selenium Architecture.
- What is TestNG?
- Installing TestNG in Eclipse
- TestNG Annotations
- Understanding usage of annotations
- Running a test in testing
- TestSuite in TestNG
- Setting priority of execution for test cases
- Skipping Tests
- Parameterizing Tests – DataProvide
- Putting Dataproviders for multiple tests in a single file
- Assertions/Reporting Errors
- HardAssertions
- SoftAssertions
- TestNG Reports
- What is ANT?
- Downloading and Configuring ANT
- Build.xml configuration
- XSLT Report Generation

Release Notes

B. TECH CSE with Specialization in DevOps

Semester Six -Year 03

Release Components.
Facilitator Guide, Facilitator Course Presentations, Student Guide, Mock exams and relevant lab guide.

Current Release Version.
1.0.0

Current Release Date.
19 Jan 2020

Course Description.
Xebia, has been recognized as a leader in DevOps by Gartner and Forrester and this course is created by Xebia to equip students with set of practices, methodologies and tools that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

The language used in this course is US English. Our sources of reference for grammar, syntax, and mechanics are from The Chicago Manual of Style, The American Heritage Dictionary, and the Microsoft Manual of Style for Technical Publications.

Bugs reported	Not applicable for version 1.0.0
Next planned release	Version 2.0.0 Jan 2021