**.How many xml files did you have in your project?can we have more than one?**

two yes

**2.How do you ensure if your test is successful or not? (ans:by using assert/ verify)**

3.How do you do data driven testing?

@dataProvider

@parameter

**.How do you run test cases parallely?**

sute level we give parallel ="class" thread count =3

Advantages of testNG over junit?



**6.can we have dependency among tests? explain with an example**

yees we can have

@test(dependsOnMethods={"login"})

method name :

**7.how do you run only the failed test cases**

**failed test case in output**

TestNG. In order achieve this we have to first understand the org.testng.IRetryAnalyzer interface. The interface definition is

public interface IRetryAnalyzer {

/\*\*

\* Returns true if the test method has to be retried, false otherwise.

\*

\* @param result The result of the test method that just ran.

\* @return true if the test method has to be retried, false otherwise.

\*/

public boolean retry(ITestResult result);

}

public interface IRetryAnalyzer {

/\*\*

\* Returns true if the test method has to be retried, false otherwise.

\*

\* @param result The result of the test method that just ran.

\* @return true if the test method has to be retried, false otherwise.

\*/

public boolean retry(ITestResult result);

}

This interface has only one method

public boolean retry(ITestResult result);

1

public boolean retry(ITestResult result);

This method will be called once a test method fails. You can get the details of the test from ITestResult input argument to this method, as shown in the method definition above. This method implementation should return true if you want to re-execute your failed test and false if you don’t want to re-execute your test. Usually the implementation of this interface decides on how many times to retry a failed tests based on a fixed counter or a complex logic based on your requirements. A simple implementation of this interface looks like this

package Tests;

import org.testng.IRetryAnalyzer;

import org.testng.ITestResult;

public class RetryAnalyzer implements IRetryAnalyzer {

int counter = 0;

int retryLimit = 4;

/\*

\* (non-Javadoc)

\* @see org.testng.IRetryAnalyzer#retry(org.testng.ITestResult)

\*

\* This method decides how many times a test needs to be rerun.

\* TestNg will call this method every time a test fails. So we

\* can put some code in here to decide when to rerun the test.

\*

\* Note: This method will return true if a tests needs to be retried

\* and false it not.

\*

\*/

@Override

public boolean retry(ITestResult result) {

if(counter < retryLimit)

{

counter++;

return true;

}

return false;

}

}

package Tests;

import org.testng.IRetryAnalyzer;

import org.testng.ITestResult;

public class RetryAnalyzer implements IRetryAnalyzer {

int counter = 0;

int retryLimit = 4;

/\*

\* (non-Javadoc)

\* @see org.testng.IRetryAnalyzer#retry(org.testng.ITestResult)

\*

\* This method decides how many times a test needs to be rerun.

\* TestNg will call this method every time a test fails. So we

\* can put some code in here to decide when to rerun the test.

\*

\* Note: This method will return true if a tests needs to be retried

\* and false it not.

\*

\*/

@Override

public boolean retry(ITestResult result) {

if(counter < retryLimit)

{

counter++;

return true;

}

return false;

}

}

We now have a simple implementation of IRetryAnalyzer. Taking a closer look at the retry method implementation, this method make sure that a failed test is retried 4 times. This is because we have specified retryLimit = 4;

Lets see how we can use it, there are two ways to include retry analyser in your tests

By specifying retryAnalyzer value in the @Test annotation

By adding Retry analyser during run time by implementing on the of the Listener interfaces

Specifying retryAnalyzer attribute in the @Test annotation

We can do this by simply using following syntax to **@Test(retryAnalyzer=”IRetryAnalyzer Implementing class”)**. Below is the code to do that

import org.testng.Assert;

import org.testng.annotations.Test;

public class Test001 {

@Test(retryAnalyzer = Tests.RetryAnalyzer.class)

public void Test1()

{

Assert.assertEquals(false, true);

}

@Test

public void Test2()

{

Assert.assertEquals(false, true);

}

}

import org.testng.Assert;

import org.testng.annotations.Test;

public class Test001 {

@Test(retryAnalyzer = Tests.RetryAnalyzer.class)

public void Test1()

{

Assert.assertEquals(false, true);

}

@Test

public void Test2()

{

Assert.as

What are the different ways you are aware of running the xml file? (ans: 1. through IDE, Through build tool-maven or ant, through cmd)

**9.have you prioritized testscripts in your projects?**

yes

@test(priority=1)

**0.Can you convert junit test scripts to testing?how?**

reate JUnit Test Case Class

Create a java class, which is a JUnit test class, TestJunit.java in C:\>TestNG\_WORKSPACE.

import org.junit.Test;

import static org.testng.AssertJUnit.assertEquals;

public class TestJunit {

@Test

public void testAdd() {

String str = "Junit testing using TestNG";

AssertEquals("Junit testing using TestNG",str);

}

}

Now, let's write the testng.xml in C:\>TestNG\_WORKSPACE, which would contain the <suite> tag as follows −

<?xml version = "1.0" encoding = "UTF-8"?>

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

<suite name = "Converted JUnit suite" >

**<test name = "JUnitTests" junit="true">**

<classes>

<class name = "TestJunit" />

</classes>

</test>

</suite>

To execute the JUnit test cases, define the property junit="true" as in the xml above. The JUnit test case class TestJunit is defined in class name

For JUnit 4, TestNG will use the org.junit.runner.JUnitCore runner to run your tests.

Compile all java classes using javac.

C:\TestNG\_WORKSPACE>javac TestJunit.java

Now, run testng.xml, which will run the JUnit test case as TestNG.

C:\TestNG\_WORKSPACE>java -cp "C:\TestNG\_WORKSPACE:C:\TestNG\_WORKSPACE\lib\junit-4.11.jar" org.testng.TestNG testng.xml

Here, we have placed the junit-4.11.jar under C:\TestNG\_WORKSPACE\lib\junit-4.11.jar.

Verify the output.

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

Converted JUnit suite

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

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

11.How can you get to know from the reports generated that which group of test cases are executed and which are not? (ans: using include and exclude tags under grouptags in testing.xml file)

<?xml version = "1.0" encoding = "UTF-8"?>

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

<suite name = "Suite1">

<test name = "test1">

<groups>

<run>

<include name = "functest" />

</run>

</groups>

<classes>

<class name = "GroupTestExample" />

</classes>

</test>

</suite>

@Test(groups = { "functest", "checkintest" }) yhis run

@Test(groups = { "checkintest" })

**12.Have you worked with listeners? Explain**

listener listens to the event defined in the selenium script and behvae acccordingly .it is used in selemiun by using listernet interface

many listerner in testng

ITESTListener

onStart

onTestSuccess

onTestFailure

onFinish

**13.How can you ignore a testscript in testing and junit?**

@**Ignore**("Test is ignored as a demonstration")

@Test

public void testSame() {

assertThat(1, is(1));

}

@Test( enabled=false )

**14.Difference between hard assert and soft assert?**

hard assert

it will terminate tc if conditions are meet

depenednet

softassert

even if condition is not matching also it will continue with exxecution

depenednet

It is independent tc

**5.Why testing with selenium?**

hy Use TestNG with Selenium? Default Selenium tests do not generate a proper format for the test results. ... Using testng, you can execute multiple test cases on multiple browsers, i.e., cross browser testing. The testing framework can be easily integrated with tools like Maven, Jenkins, etc.

open source

cross broswer testingg

selenium is use to automate web application .seelelnium interact with web broswer and perform functionality

**Have you worked with logs in testing? how do you achieve it?**

The above class extends TestListenerAdapter, which implements ITestListener with empty methods.

mport org.testng.ITestResult;

import org.testng.TestListenerAdapter;

public class CustomListener extends **TestListenerAdapter**{

private int m\_count = 0;

@Override

public void onTestFailure(ITestResult tr) {

log(tr.getName()+ "--Test method failed\n");

}

@Override

public void onTestSkipped(ITestResult tr) {

log(tr.getName()+ "--Test method skipped\n");

}

@Override

public void onTestSuccess(ITestResult tr) {

log(tr.getName()+ "--Test method success\n");

}

private void log(String string) {

System.out.print(string);

if (++m\_count % 40 == 0) {

System.out.println("");

}

}

}

<?xml version = "1.0" encoding = "UTF-8"?>

<suite name = "Simple Logger Suite">

**<listeners>**

**<listener class-name = "CustomListener" />**

**</listeners>**

<test name = "Simple Logger test"> l parallel sute

group below test and paramater also

listnere after suite

<classes>

<class name = "SampleTest" />

</classes>

</test>

</suite>

17.what is the difference between @before test and @before method as both gets executed before the tests?

@BeforeMethod: The annotated method will be run before each test method.

@BeforeTest: The annotated method will be run before any test method belonging to the classes inside the <test> tag is run

18.Have you worked with junit?

JUnit is a unit testing framework for Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit, that originated with JUnit.

JUnit is an open source Unit Testing Framework for JAVA. It is useful for Java Developers to write and run repeatable tests.

Behavior Driven testing is an extension of TDD. Like in TDD in BDD also we write tests first and the add application code. T

As the name implies, it is used for Unit Testing of a small chunk of code.

.Difference between assert and verify?

20.Is assert from testing class or selenium class?

testng

21.How do you do cross platform testing parallely? (ans:Selenium Grid with testing)

2.What is testing?

estNG is an automation testing framework in which NG stands for "Next Generation". TestNG is inspired from JUnit which uses the annotations (@).

Using TestNG you can generate a proper report, and you can easily come to know how many test cases are passed, failed and skipped.

You can execute failed test case separately. For example.

What is the sequence of these annotations?

BeforeSuite

BeforeTest

BeforeClass

BeforeMethod

Test

AfterMethod

AfterClass

AfterTest

AfterSuite

.How do you generate reports in testing? the steps?

The TestNG will generate the default report.

When you execute testng.xml file, and refresh the project. You will get test-output folder in that folder.

Right click on the emailable-report.html and select the option. Open with the web browser.

In this tutorial, you will learn-

Method-1: emailable-report.html

Method-2: index.html

Method-3: Reporter Class

.what are the annotations used in your project? Why do you use them?

@beforeclass

@beforetest

@before method

7.Example of assert statements used in your project?

Assert.equals(expectd,actual con);

Assert.asertTrue

.When Smoke test fails, how do you stop testing other test cases automatically? (ans: using dependency in testing, depend on Groups in @test)

29.How do you do cross browser testing?

parameter

selenium gri d

How do run or execute your scripts ?

click run as testng

or by tetsng.xml run as testng suite