**TESTNG(https://www.softwaretestingmaterial.com/testng-interview-questions)**

It is an open-source [automation testing](https://www.scientecheasy.com/2020/07/selenium-automation-testing.html/) framework based on JUnit (in Java) and NUnit (in C#).

TestNG is an advanced framework that can be integrated with Selenium or any other automation tool.

**Why do we use TestNG Framework with Selenium?**

1. Selenium WebDriver does not generate detailed test or summary reports. If Selenium test cases are created by integrating TestNG with Selenium WebDriver, no external task is required to generate detailed test reports.
2. We can execute selective test cases by integrating TestNG with Selenium WebDriver.
3. Using TestNG, we can also execute failed test cases separately.
4. We can easily integrate the TestNG framework with various tools and plugins like ANT, Maven, Jenkins, Integrated Development Environment (Eclipse), etc.
5. Verification can be achieved with the help of only TestNG.
6. We can organize test cases easily in a proper way.

**Features of TestNG Framework**

**1. Object-oriented:** Since TestNG is written in Java, it uses more object-oriented and Java features.

**2. Multiple Before/After annotations:** TestNG offers multiple types of Before/After annotations to create test cases.

Annotations are a kind of instructions for the defined set of code that instructs how to run.

**3. Priority:** It is one of the major features provided by TestNG where we can instruct TestNG to execute the dependent test method after running the main test method by setting priority.

Using priority, we can execute test methods in TestNG in a different order. We can also set priority whether the dependent test method has to be executed or not in case the earlier test method fails.

**4. Groups/group of groups:** This feature allows us to assign certain test methods into particular named groups. Using this feature, we can execute multiple test cases into a particular group.

**5. XML file:** An XML file is a file that is used to configure test suites in TestNG using classes, test methods, and packages.This file is also used to run the whole test from a single place by passing the parameters to the test methods or classes.

**6. Parameterization of test methods:** This feature allows users to pass the parameter values directly through XML file to the test methods used in tests.

**7. Data-driven testing:** This feature is widely used in test suites that allow users to execute the same test method multiple times based on different sets of data.

**8. Multi-threaded execution:** This feature is mainly used to execute test cases in a multi-threaded environment. Using this feature, we can execute multiple test cases in parallel to reduce the execution time.

**9. Test report:** TestNG internally generates an XML and HTML test report in a readable format by default for its test execution.

Generation of test report includes a number of test cases run, passed, failed, and skipped. You can also add custom reports if required.

**10. Main method:** There is no need for the static main method in the TestNG. The method need not be static.

**11. Inbuilt Exception handling:** TestNG has an inbuilt [exception handling mechanism](https://www.scientecheasy.com/2020/08/exception-handling-in-java.html/) that allows us to run the program without terminating unexpectedly.

**12. Thread-safe:** TestNG framework supports a thread-safe environment when executing more than one [thread](https://www.scientecheasy.com/2020/08/thread-in-java.html/).

**Advantage of TestNG Framework over Junit**

1. TestNG generates XML, HTML test reports in a readable format. It provides a very good reporting structure.

2. Test method can be grouped and prioritized more easily which is not possible in JUnit. We can easily execution patterns.

3. Annotations of TestNG are easy to understand for creating test cases. It provides superior and advanced annotations over JUnit.

4. We can set the dependencies of Test cases.

5. Parallel testing is also possible in TestNG.

**Disadvantage of TestNG Framework**

The main disadvantage of TestNG framework is that it does not allow us to define our own reporting format.

**What is Junit?**

JUnit is an open source unit testing tool and used to test small/large units of code.

To run the JUnit test you don’t have to create a class object or define the main method.

[JUnit](http://junit.org/) provides assertion library which is used to evaluate the test result.

Annotations of JUnit are used to run the test method.

**What is Annotation?**

Annotation is the set of code that controls how the method below them has to be executed. It determines the order of execution.

It helps to run our tests in an organized manner

* **Suite:** A suite is made of one or more tests.
* **Test:** A test is made of one or more classes.
* **Class:** A class is made of one or more methods.

**Three sections of annotations:**

**Precondition annotations:** @BeforeSuite, @BeforeTest, @BeforeClass, @BeforeMethod

**Test annotation:** @Test

**Postcondition annotations:** @AfterSuite, @AfterTest, @AfterClass, @AfterMethod

**TESTNG lifecycle:**

@BeforeSuite

This annotated method will be run before running all the tests in the entire suite.

A method annotated with @BeforeSuite will be executed only once.

@AfterSuite

This annotated method will be run after running all the tests in this suite.

A method annotated with @AfterSuite will be executed only once.

@BeforeTest

Method annotated with @BeforeTest will be executed before running each unit test.

It will be executed only once, before executing any test method within the same test class.

@Aftertest

Method annotated with @AfterTest will be executed after running each unit test.

It will be executed only once, after finishing the execution of all test methods within the same test class.

@Before Class

This annotated method will be executed only once before any of the tests in the current class are run.

It can be used to set up the configuration and initialization which is common to all test methods in the current class.

@AfterClass

This annotated method will be executed only once after finishing all the tests of that current class.

@BeforeMethod

Method annotated with @BeforeMethod will be executed before each method annotated with @Test.

It can be used to set that data that is repeating before each @Test annotated method.

@AfterMethod

Method annotated with @BeforeMethod will be executed after each and every method annotated with @Test.

@Test

Methods annotated with @Test are called test methods that serve as a unit test.

In @Test methods, we will write the logic of the application which we want to automate.

We can pass attributes to our test method.The output depends on these test methods for success or failure.

@DataProvider

It marks a method as a data providing method for a test method.

It returns an object double array(Object [ ][ ]) as data to the test method.

@Parameters

This @Parameters annotation is used to pass parameters to a test method.

The values of parameters are provided using the testng.xml file at runtime.

This parameter annotation can be used with any of the Before/After, Test annotated methods.

DATAPROVIDER

**Data Provider in TestNG** is a method used when a user needs to pass complex parameters.

Complex Parameters need to be created from Java such as complex objects, objects from property files or from a database can be passed by the data provider method.

The method is annotated by @DataProvider and it returns an array of objects.

**Steps to read from excel**

* Open the workbook using filename and path
* Go to specific sheet(by sheet name or index)
* Go to specific row(by index)
* Go to specific column (by index) -cell
* Read the content

XSSFWorkbook- Workbook- HSSFWorkbook

XSSFSheet- Sheet- HSSFSheet

XSSFRow- Row- HSSFRow

XSSFCell- Cell- HSSFCell

Cell Type:

XSSFCell.CELL\_TYPE\_STRING

XSSFCell.CELL\_TYPE\_NUMERIC

XSSFCell.CELL\_TYPE\_BOOLEAN

XSSFCell.CELL\_TYPE\_BLANK

XSSFWorkbook wbook = new XSSFWorkbook(“data/CreateLead.xslx”);

XSSFSheet sheet = wbook.getSheetAt(0);

int rowCount = sheet.getLastRowNum();

short columnCount = sheet.getRow(0).getLastCellNum();

for(int i=1;i<=rowCount;i++)

{

XSSFRow row =sheet.getRow(i);

For(int j=0;j<columnCount;j++)

{

XSSFCell cell = row.getCell(j);

System.out.println(cell.getStringCellValue());

}

Wbook.close();

WHAT IS APACHE POI:

It is an open source java based library to handle Microsoft office documents by using java based programming language

**WHAT IS POM?**

**Page object model (POM)** is an object design pattern that is used to create an object repository for web UI elements.

This kind of design pattern is used in any kind of framework like Data-Driven, Modular and Hybrid framework

In this model, we create a separate corresponding page class for each web page in the application and store all locators to inspect elements on the page, respective methods to interact with those elements, and variables to use them.

**COMPONENT OF POM: Page Class + Test Class**

Each page class will locate the WebElements of the corresponding web page and perform operations on those WebElements by methods.

The test classes use methods of this page object class whenever they need to perform a test with the UI of that page.

**NEED FOR POM**

If the UI of webpage some changes in the future then the tests need not to be changed, only the code within the page class needs to be changed.

Therefore, locators and test scripts are stored separately in the Selenium page object model.

**ADVANTAGES OF POM**

**Code Reusability:** POM can achieve code reusability by witing code and using it in different test

**Code Maintainability:** If there is a change in the web element, we will need to make the change in just 1 page class and not in the 10 different scripts.

**Object Repository:** Pages can be defined in an interface like members

**Readability:** POM can help in improving readability by clean separation through Test code and Page Specific code