TestNG

A powerful testing framework in Java

TestNG Intro

What is testNG?

TestNG is an open sourced testing framework in Java that is used for Unit, Integration and end-to-end testing of Java applications. It is inspired by the JUnit and NUnit but introduced many new features.

Some of the features include:

- Parallel execution
- More annotations
- Groups
- Data Driven Testing
- Reporting
- . . .

Assertions

Hard Assertions

- assertEquals() / assertNotEqual()
- assertTrue() / assertFalse()
- assertNull() / assertNotNull()

Soft Assertions

- **assertAll()**

Differences between Hard and Soft assertions

Hard assertions will fail the test method immediately when an assertion fails, the remaining assertions will not be executed. Soft assertions, instead, will continue with the remaining assertions and reporting all the failures at the end.

Annotations and Execution Flow

Terminologies:

- Test Method / Case: Data + execution step + Assertion
- Test Execution: an execution that contains different test cases
- Suite: Nothing but a collection of tests

Annotations:

- @Before/AfterSuite
- @Before/AfterTest
- @Before/AfterClass
- @Before/AfterMethod
- @Test: annotate a test method

TestNG.xml

What is TestNG.xml?

It is a xml file that can be used to configure your test suite.

Why do we need an external xml file for configuration?

In the runtime environment, it's impossible to change the code directly, so we need to decouple the parameters and configurations from the source code.

How to write xml file?

See the code demo.

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="demo suite">
    <test name="test1">
        <classes>
            <class name="class1">
                <methods>
                    <include name="method1"/>
                </methods>
            </class>
        </classes>
    </test>

    <test name="test2">
        ...
    </test>
</suite>
```

Parameterization

What is parameterization in testing?

Parameterization in testing is the process of using variables or parameters in test cases instead of hard-coded values.

Why do we need parameterization?

It is a way to make test cases more flexible and reusable.

How to perform parameterization in TestNG?

Using <parameter> tag in the xml and @Parameter annotation in the class.

Test Prioritization

What is Test Prioritization?

It is the process of assigning different priorities to test methods in order to establish the execution flow.

Why do we need Test Prioritization?

By default, the test methods are executed in alphabetical order in TestNG, but sometimes we need to specify our own order of execution.

How do we perform Test Prioritization?

Set “priority” property in the @Test annotation

Test Dependencies

What are test dependencies in TestNG?

A way to describe the relationships between test methods.

Why do we need test dependencies in TestNG?

- Real life examples of order management in Amazon.com
- Issues with priorities (See the code demo)

How to specify the test dependencies in TestNG?

We have method, class, group level dependencies, details is shown in the code demo.

Test Grouping

What is grouping in TestNG?

A way to categorize different test methods.

Why do we need grouping in TestNG?

For better and easier management of your test methods and class. Just like tags in leetcode questions, a test method can have multiple groups and a group can have multiple test methods.

How to perform grouping in TestNG?

By setting the “groups” property in the @Test Annotation, we can also create our meta groups inside the xml file.

Test Ignore

What is ignore in TestNG?

Ignore some test cases during the Test Discovery Stage

In **TestNG** we have different levels of ignoring:

- **Package Level**
- **Class Level**
- **Method Level**

Data Provider

What are data providers in TestNG?

They are methods that can be used to provide test data to test method.

Why do we need data providers in TestNG?

- Decouple the test data from test methods, so our testing framework is more flexible and reusable.
- Improve modularity, make it easy to maintain and develop.

How to create and use our own data providers in TestNG?

We can use @DataProvider to define a DataProvider

Parallel Execution

Background knowledge Required:

MultiThreading in Java MultiThreading  Parallel Execution !!!

Real life examples:

- MultiThreading but not parallel execution: Playing chess requires two threads, but you and your opponent cannot move at the same time.
- Parallel Execution: Your RTX 4090 can have at most one million threads running simultaneously to render the graphics of the game.

Parallel Execution

What is Parallel Execution in TestNG?

Execute your tests in parallel.

Why do we need that?

Speed your testing process, optimize your testing framework.

How do we perform parallel execution?

- Method Level: xml file, Dataprovider, Method Invocation
- Class Level , Test Level: xml file

TestNG Listeners

What are TestNG Listeners?

They are a set of interfaces in TestNG framework which can listen to different types of events and Life Cycles in testNG, for example ITestListener,

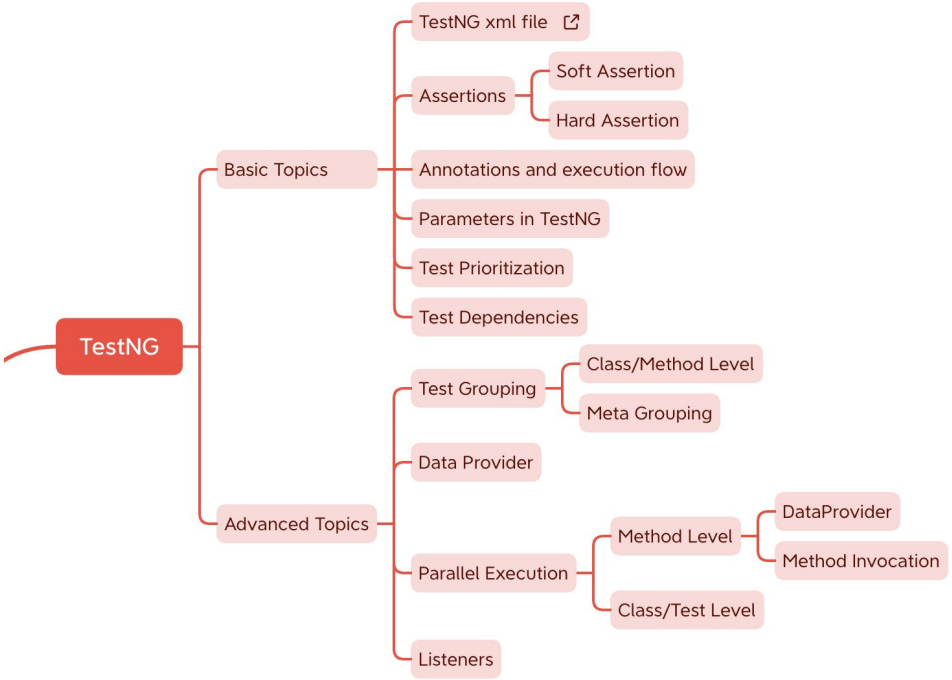
Why do we need TestNG listeners?

Customize the behavior of TestNG during test execution. For example reporting and error handling.

How do we use these TestNG listeners?

Demo: Attaching screenshot of browser to TestNG reports when there are failures detected in the test.

Summary


TestNG
Basic Topics
TestNG xml file
Assertions
Soft Assertion
Hard Assertion
Annotations and execution flow
Parameters in TestNG
Test Prioritization
Test Dependencies
Advanced Topics
Test Grouping
Class/Method Level
Meta Grouping
Data Provider
Parallel Execution
Method Level
DataProvider
Method Invocation
Class/Test Level
Listeners