**TestNG - Dependent Tests**

## What Are Dependent Test In TestNG?

Often, we want to run our test cases in a particular order in TestNG. We may use the [***priority parameter***](https://www.toolsqa.com/testng/testng-annotations/) for that, no doubt, but priority will run all the cases without looking for the relationship we want to define (alphabetically for the same priority). The dependent tests in TestNG determine the dependency of a test on a single or group of tests. In this case, we say that a test is dependent on another test. It is similar to saying a browser is dependent on the internet. No internet means no purpose in running the browser. Providing dependencies among tests also helps us in the scenarios where we want to share the state or data between the methods. So, first things first, how do we specify these dependent tests in TestNG?

### ***How To Use dependsOn attribute in TestNG?***

TestNG allows you to specify dependencies in the following two ways:

* ***Using attributes dependsOnMethods in @Test annotations***

The dependsOnMethods lets us make a test depend on a particular method. For example, look at the following code:

public class Dependent {

@Test (dependsOnMethods = { "OpenBrowser" })

public void SignIn() {

System.out.println("This will execute second (SignIn)");

}

@Test

public void OpenBrowser() {

System.out.println("This will execute first (Open Browser)");

}

}

Here, our method SignIn() is dependent upon the method OpenBrowser()

* ***Using attributes dependsOnGroups in @Test annotations***.

The dependsOnGroups attribute lets us make a test depend on a whole group rather than a single test. For example, see the code below:

public class GroupDependency

{

@Test(dependsOnGroups = { "SignIn" })

public void ViewAcc() {

System.out.println("SignIn Successful");

}

@Test(groups = { "SignIn" })

public void LogIn() {

System.out.println("Logging In Success");

}

}

Here, our test method, "***ViewAcc()***", depends upon the group "***SignIn***".

Refer [***TestNG Annotations***](https://www.toolsqa.com/testng/testng-annotations/) to know more about the annotations, their benefits, and how they work in the hierarchy with different possible scenarios.

Let us see the different ways in which we can provide the dependency in TestNG.

### ***Single Dependent Test Methods In TestNG***

A single dependent test in TestNG is declared when a single test depends on another test. It is similar to the example we saw above. We use dependsOnMethods for the same purpose.

Take a look over the below example:

import org.testng.annotations.Test;

public class DependsOnTest {

@Test (dependsOnMethods = { "OpenBrowser" })

public void SignIn() {

System.out.println("User has signed in successfully");

}

@Test

public void OpenBrowser() {

System.out.println("The browser is opened");

}

@Test (dependsOnMethods = { "SignIn" })

public void LogOut() {

System.out.println("The user logged out successfully");

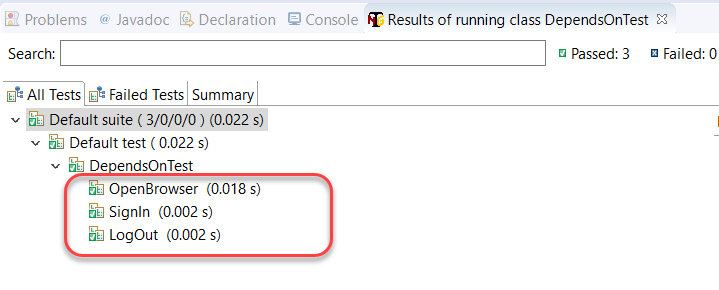
}

}

The above code contains three functions:

* ***SignIn()*** - This function depends on the method OpenBrowser (depicted by dependsOnMethod).
* ***OpenBrowser()*** - This method does not depend on any method; hence it is a standard test case.
* ***LogOut()***-  This method depends on the method SignIn() (depicted by dependsOnMethod).

If you think about it, it is helping us in testing in a more significant way. By making one test depend upon another test, we can ensure that SignIn happens after the browser has opened. Logout happens only when the SignIn i. You can use these kinds of logic queries to bring more out of TestNG in your tests.

Let's verify the output and see if it matches our expectations.

The result is as expected. It brings us to our next question whether we can make one test rely on more than one test? Let's explore in the next section.

### ***Multiple Dependent Tests In TestNG***

A single test depends on multiple tests in TestNG. We make use of defining various dependencies for the same. Observe the code below:

import org.testng.annotations.Test;

public class DependsOnTest

{

@Test

public void OpenBrowser() {

System.out.println("Opening The Browser");

}

@Test(dependsOnMethods = { "SignIn", "OpenBrowser" })

public void LogOut() {

System.out.println("Logging Out");

}

@Test

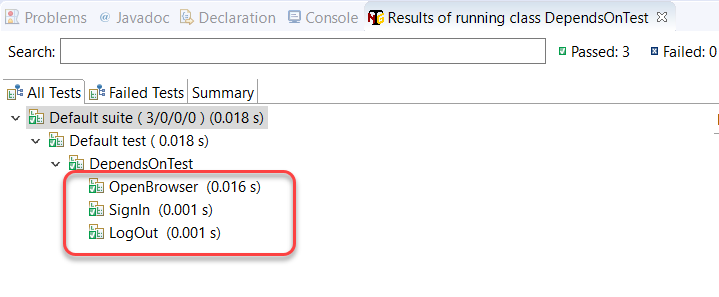
public void SignIn() {

System.out.println("Signing In");

}

}

In the above example, I have created multiple dependencies by making the test method "***LogOut***" depend upon two different tests, which are "***SignIn***" and "***OpenBrowser***". The Logout() method has been moved to the middle intentionally to demonstrate the use of dependent tests in TestNG. Logically, it would come at the last.

The above code will give the following output in Eclipse:

As expected, the LogOut() method executed after the methods on which it depended has executed.

### ***Inherited Dependent Test Methods In TestNG***

Inheritance is a popular concept in object-oriented programming paradigms, and especially if you have worked in Java, you must have heard and used it a lot. In the previously used methods, we used all the tests of the same class in TestNG. ***In inherited dependent test methods in TestNG, we create dependency among the methods that belong to different classes, and one of the classes inherits the functionalities of another class***.

import org.testng.annotations.Test;

class SuperClass

{

@Test

public void OpenBrowser() {

System.out.println("BrowserOpened");

}

}

public class InheritedDependencyTest extends SuperClass

{

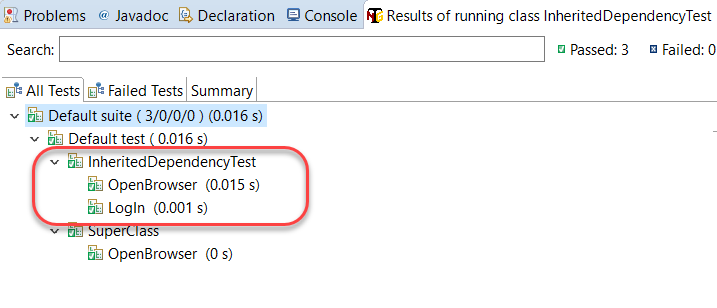
@Test(dependsOnMethods = { "OpenBrowser" })

public void LogIn() {

System.out.println("Logged In");

}

}

As seen in the above code, the InheritedDependencyTest class inherits the SuperClass and hence can depend on any function described in SuperClass. Consequently, the above code will result in the following output:

Please note that the name of the TestNG class is InheritedDependencyTest, which is the base class in the above code. As a practice, you can swap superclass and base class names to analyze the response from TestNG.

## What Are Group Dependent Test?

In the above sections, we just created dependencies of a single test over another test (single or multiple). While creating dependencies in multiple tests, we have to write each test name separately manually. For example, if I want to make a test depend on three other tests, I need all the three names in the dependsOnMethod attribute. What if we want to create a dependent test that depends on a group of tests? If we could do that, we can have a group of a test run before this test and then we just need to mention a single group name instead of multiple test names.

Fortunately, ***TestNG provides the functionality of making a test depend on a group of tests***. Let's see the following code to understand the same.

import org.testng.annotations.Test;

public class GroupDependency

{

@Test(dependsOnGroups = { "SignIn" })

public void ViewAcc() {

System.out.println("View Your Dashboardd");

}

@Test(groups = { "SignIn" })

public void OpenBrowser() {

System.out.println("Browser Opened Successfully");

}

@Test(groups = { "SignIn" })

public void LogIn() {

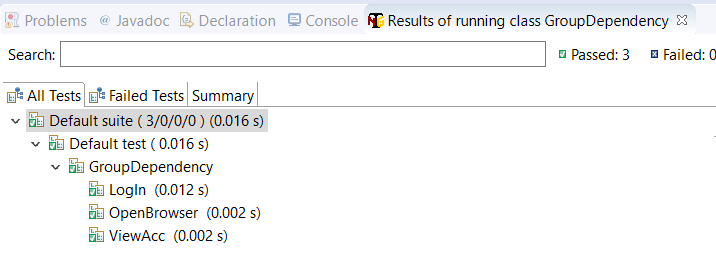
System.out.println("Login Into The Account");

}

}

In the above code, we created a group called SignIn and a test method ViewAcc that depends on that group.

The group SignIn contains two tests under it called OpenBrowser (describing the successful event of opening a browser) and LogIn (describing the successful login event).

Run the file to see that the ViewAcc will run in the last.

As expected, the ViewAcc runs after all the group tests have run on which it depended.

But, notice the tests run in the group SignIn. We intended to run the OpenBrowser test first and LogIn after that. I mean, that makes logical sense. But in the above run, LogIn has run before the OpenBrowser method. That would create problems as how would you LogIn if the browser has not yet opened.

So, any guesses why that happened?

***Since the group and the annotation is the same, TestNG will run the methods in the alphabetical order***. Try renaming the LogIn as something like Testing(). Since T comes after O, OpenBrowser will run first. For more on this, please refer to [***TestNG Annotations and its hierarchy***](https://www.toolsqa.com/testng/testng-annotations/).

## TestNG Dependent Test In XML Suite

All that we discussed in the above sections of this tutorial was somehow related to the changes done in the TestNG test case file. In this section, we will move our dependency commands over to the XML file.

***TestNG lets you create dependencies between groups in the XML file***. So, if you have multiple groups in the TestNG file, you can create the dependent tests in between them in the XML file.

Let's create multiple groups in our TestNG test case file first before jumping to the XML file.

import org.testng.annotations.Test;

public class GroupDependency

{

@Test(groups = { "viewacc" })

public void ViewAcc() {

System.out.println("View Your Dashboardd");

}

@Test(groups = { "openbrowser" })

public void OpenBrowser() {

System.out.println("Browser Opened Successfully");

}

@Test(groups = { "login" })

public void LogIn() {

System.out.println("Login Into The Account");

}

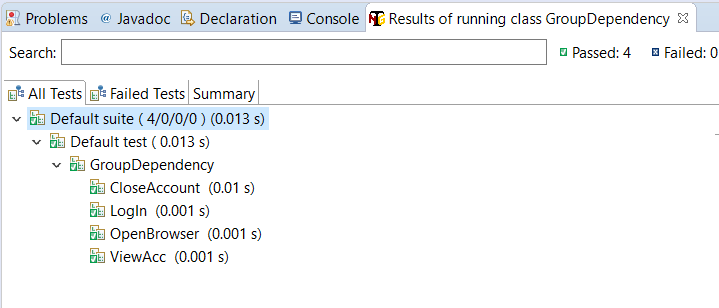
@Test(groups = {"logout"})

public void CloseAccount() {

System.out.println("Closing The Account");

}

}

In the above code, I have created four groups and declared one method in each of them. Notice that there are no dependencies in this file so the file will currently run the methods in the alphabetical order as follow:

We don't want to close our account before logging in.

Let's hop on to the XML file and create some dependencies.

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

<suite name="TestNG XML Dependency Suite" >

<test name="ToolsQA" >

<groups>

<dependencies>

<group depends-on= "openbrowser" name= "login"></group>

<group depends-on= "login" name= "viewaccount"></group>

<group depends-on= "viewaccount" name= "logout"></group>

</dependencies>

</groups>

<classes>

<class name="GroupDependency" />

</classes>

</test>

</suite>

In the "***dependencies***" tag, I have created the flow of groups that I want to execute. There are a total of three components in each dependency that I have created.

***<group>*** - The tag you need to specify to tell XML that we are talking about the groups.

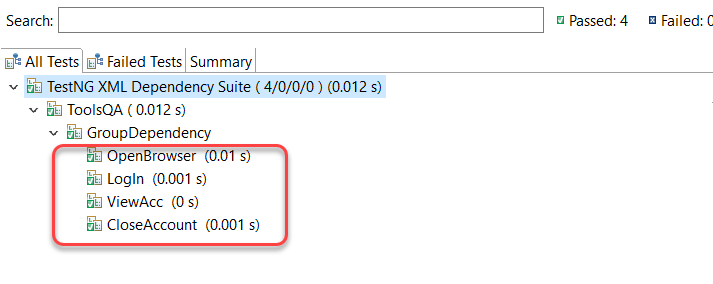
***depends-on*** -\* The name of the group on which you want this group to depend\*.

***name***- Name of the group that you want to depend on.

*<group depends-on = "\*\*\*openbrowser\*\*\*" name = "\*\*\*login\*\*\*"></group>*

***The above code states that the group named "login" should depend upon the group with the name "openbrowser".***

Similarly, you can analyze the other two dependencies in the above XML code.

Running the test suite will give us the following output:

Since before CloseAccount was running before every other test, this time, we have run it in the correct order by creating the dependency.

I hope that was a clear and crisp tutorial about the dependent tests in TestNG. Till this point, we have covered a lot by which it would be straightforward to write good tests in [***TestNG***](https://testng.org/doc/) and use different types of tests also. Practice your way as much as you can to write good tests in TestNG.