Rex Jones II

Chapter 3: TestNG Annotations

Table of Contents

Chapter 3.1: TestNG Annotations Overview

Introduction

Hello and Welcome To Chapter 3, TestNG Annotations. This chapter is a series of 3 videos: An Overview of TestNG Annotations, Introduction To TestNG Annotations, TestNG Execution Flow, and we will view the TestNG xml File.

Chapter 3 Overview

This video is 3.1 an Overview of TestNG Annotations. Video 3.2 is an Introduction To TestNG Annotations. In that video, we will talk about What Are TestNG Annotations, How To Add TestNG Annotations, and the Configuration Annotations. Video 3.3 is a video about the Execution Flow of the TestNG Annotations and the TestNG xml File. Our execution flow depends on the annotations and the xml file helps understand the TestNG Suite, Test, Class, and Methods.

Next is video 3.2, an Introduction To TestNG Annotations.

Chapter 3.2: Introduction To TestNG

Introduction

Welcome To Chapter 3.2, Introduction To TestNG Annotations. In this chapter, we will discuss What Are TestNG Annotations, How To Add TestNG Annotations, and the TestNG Configuration Annotations.

What Are TestNG Annotations

What Are TestNG Annotations? A TestNG Annotation is data that have a special meaning for a Java method. It provides information about how the annotation will control the execution order. I see TestNG Annotations as Pre-Conditions, Conditions, and Post-Conditions. Why? Because our Automation Test Scripts are very similar to Manual Test Cases. Sometimes with Manual Test Cases, there are Pre-Conditions that must be set up before we start our test. Our test is the Condition and after our test is the Post-Condition.

It's the same with automation, we have Pre-Conditions that must be set up before we test then we have our test. Our test is the Condition. Next is the Post-Condition that is performed after we complete our testing. Notice the @ symbol. In some places, you may hear @ symbol or @ sign. Both names are short for at a rate which is an accounting term. This symbol is placed in front of each TestNG Annotation. All of the Pre-Conditions begin with @Before while the Condition is @Test. @Test is a key annotation because it performs our test. Last, we have the Post-Conditions that all start with @After.

How To Add TestNG Annotations

How To Add TestNG Annotations? Let's go to Eclipse. There is more than 1 way to add an annotation. I believe most people start from scratch and write our annotation. For example, write @BeforeMethod then write our method. public void setup () and import the annotation. A short cut for importing and organizing our imports is CTRL + SHIFT + O. If you want to see a list of Eclipse shortcuts select CTRL + SHIFT + L and we see a lot of shortcuts including Organize Imports.

Another way to add TestNG Annotations is by selecting the package, right click, New then Other. Type TestNG, Select TestNG class then click Next. Do you see the Annotations? Let's select all of the annotations except for DataProvider. We are going to cover DataProvider in a subsequent chapter called Data Driven Testing. Data Driven Testing is when we run 1 Test Script with many sets of data. Change the Class Name to Configuration_Annotations and Click Finish.

Configuration Annotations

A Configuration Annotation is an annotation that begins with Before or After. They are called Configuration Annotations because the Before Annotations help us set up variables and configurations before starting test execution. The After Annotations help us clean up everything after executing our test. If we take a look at the TestNG Annotations package, we will see all of the annotations. The Configuration Annotations start at AfterClass and stop at BeforeTest.

In our Editor, the annotations are displayed in order from lowest to highest. Let's start with the highest level and add some print statements. BeforeSuite/AfterSuite run before a suite start and after all Test Methods. So, let's write BeforeSuite, Chrome - Set Up System Property. AfterSuite, Chrome – Clean Up All Cookies.

BeforeTest/AfterTest run before a test start and after all Test Methods. Now that we have set up Chrome, BeforeTest will Open Chrome and AfterTest will Close Chrome. BeforeClass/AfterClass run before a test class start and after all Test Methods. After opening Chrome, BeforeClass will Open Test Application and AfterClass will Close Test Application.

Next, we have BeforeMethod/AfterMethod which run before a Test Method and after a Test Method. Now that we finally opened the Test Application, let's Sign In and Sign Out. Our test will Search For Customer so change the name to searchCustomer. We can add as many test annotations as we want but let's add 1 more to Search For Product and change the name to searchProduct.

The annotations can be placed in any order on the editor because TestNG identifies the methods by looking up the annotation. For example, we can place BeforeSuite anywhere on this editor and it will always execute first.

Next in Chapter 3.3, we will see the order of how these annotations execute.

Chapter 3.3: TestNG Annotations Execution Flow & xml File

Introduction

Welcome To Chapter 3.3, TestNG Annotations Execution Flow and the xml File. In this chapter, we will discuss the Execution Flow of TestNG Annotations, View the TestNG xml File For TestNG Suite, Test, Class, & Method tags, then decide Which TestNG Annotations We Choose For Testing.

The TestNG Annotations dictate the execution order and the xml file is a great way to see why annotations execute in that order. We will take a look at some of the annotations to see why Test Requirements determine our TestNG Annotations.

TestNG Annotations Execution Flow

The execution flow depends on our annotations. Therefore, the methods execute according to the rank of each annotation. Let's go to Eclipse and change the annotation order.

We will change the order on the Editor to match the order in how they will show up on the Console. BeforeSuite, BeforeTest, BeforeClass, BeforeMethod, first Test Method Search Customer, second Test Method Search Product, AfterMethod, AfterClass, AfterTest, and AfterSuite. Let's Run.

We see BeforeSuite / Chrome – Set Up System Property, then we see BeforeTest / Open Chrome, next is BeforeClass / Open Test Application, and we have BeforeMethod / which is Sign In. Here's the interesting part, the BeforeMethod always run before the Test Method. In this case, the first Test Method is Search For Customer which has an @Test Annotation and the AfterMethod always run after the Test Method. The AfterMethod is Sign Out.

We finished searching for a customer. However, Chrome is still set up, Chrome is still open, and the application remains open but we are not signed into the application. Now we start over with the BeforeMethod, which is a Pre-Condition to Sign In before Searching For A Product then the Post-Condition, AfterMethod Sign Out of the application. This shows us how TestNG executes the Pre-Condition, the Condition, and the Post-Condition based on our annotations.

Next, we have AfterClass / Close Test Application, AfterTest / Close Chrome, and AfterSuite – Clean Up All Cookies. In the Console, we see both searchCustomer and searchProduct PASSED. The TestNG results tab also shows both tests PASSED.

View xml File To See TestNG Suite, Test, Class, & Methods

1 Test

View xml file. The purpose of an xml file is to store data and to carry data for all of our testing. We see a tag for suite, a tag for test, a tag for class, and a tag for methods. A Suite can have 1 or more tests. A Test can have 1 or more classes and a Class can have 1 or more methods. The xml file gives us a picture on why the annotations execute in a particular order.

Let's Run our Test Suite. We see the same output. It's the same output because this execution has 1 test tag. We see BeforeSuite, BeforeTest, BeforeClass, BeforeMethod executes before both Test Methods – Search For Customer and Search For Product, then the AfterMethod executes after both Test Methods.

Next is AfterClass, AfterTest, and AfterSuite. Do you see how the execution order is consistent with the xml file? Let's look at an xml file that has 2 test tags.

2 Tests

This xml file does not include the Test Methods together. We see the 1st Test Name is Search For A Customer and the 2nd Test Name is Search For A Product. Let's run. The output is different. This time our execution does not immediately Sign back into the application after Signing out of the application. Now, the execution processes AfterClass – Close Test Application and AfterTest – Close Chrome after signing out. Why? Because in the xml file, the Test Methods are not in included the same class. Therefore, execution will not sign back in to start our next test. Notice, Chrome is still set up after executing

BeforeSuite. That means the AfterSuite annotation is the only annotation that have not been executed. That completes searching for a customer.

Next is Search For A Product which starts with BeforeTest – Open Chrome, BeforeClass – Open Test Application, then BeforeMethod – Sign In and Search For A Product. Finally, AfterMethod – Signs Out, AfterClass – Close Test Application, AfterTest – Close Chrome, now AfterSuite – Clean Up All Cookies from Chrome.

In reality, we probably would not use 4 Before Configuration Annotations and 4 After Configuration Annotations. However, I wanted you to see how the xml file is connected to the annotations and how the xml file allows us to execute the same methods in 1 test or more than 1 test.

Which Annotation(s) Do We Choose For Testing

Which annotations do we choose for testing? The annotations we choose for testing depends on our Test Requirements. Do we need our test to set up a Pre-Condition before every Test Method and clean up with a Post-Condition after every Test Method or do we need 1 Pre-Condition before all of the Test Methods and 1 Post-Condition after all of the Test Methods? Let's walkthrough our Test Application then I will show you the difference using 2 pairs of Configuration Annotations and 3 Test Methods.

The Test Requirement is to sign in with Username Admin and Password admin123 then click the Login button. After signing in, we click the Admin tab then search for a user. Finally, we sign out.

Our code uses a BeforeMethod annotation and AfterMethod annotation. First, we setup the test, then sign into the application, after signing into the application, we search for a user and sign out. After signing out we teardown our test which is close the browser.

Let's run and see what happens. We see 2 failures in the Console. The results tab also shows 2 Failures. Go back to the Console and look at our first Test signIn. Step 1 shows we opened Chrome and the application, Step 2 shows we signed into the application, but Steps 3 and 4 are missing before Step 5 close Chrome and the application. It passed but did not perform the steps we expected.

Let's look at the next test userSearch. Step 1 Open Chrome & the application. Steps 2, 3, and 4 are missing but we see Step 5 which Close Chrome and the application. Why did the Test Script skip steps 2 through 4? It skipped those steps because we cannot open Chrome and the application then search for a user. We cannot search for a user because we have not signed into the application. The console shows Failed: userSearch. Unable to locate element. It could not locate the Admin tab to begin searching for a user.

Same with the userSignOut. Our Test Script could not sign out of the application because it never signed into the application. Steps 2, 3, and 4 are missing. If we scroll down, the console shows FAILED: userSignOut. With this Test Requirement, we must use BeforeClass/AfterClass or BeforeTest/AfterTest. So, our code does not open the application before every Test Method and close the application after every Test Method.

Now watch what happens when we use the same code but change the Configuration Annotations to BeforeClass and AfterClass. Let's Run. All 3 Test Scripts PASSED. The results tab shows all 3 PASSED. The

Console shows Step 1, we opened Chrome and the application, Step 2, we signed into the application, Step 3, we searched for a user, Step 4 we signed out of the application, and Step 5, we closed Chrome and the application.

TestNG provides a lot of annotations for our Test Requirements. This is a list of the annotations we covered in Chapter 3. Next in Chapter 4, we will see how the Test Methods execute using a priority attribute.