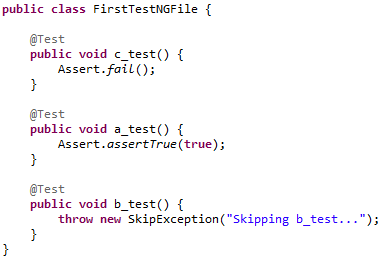
Day-47

**Annotations used in TestNG**

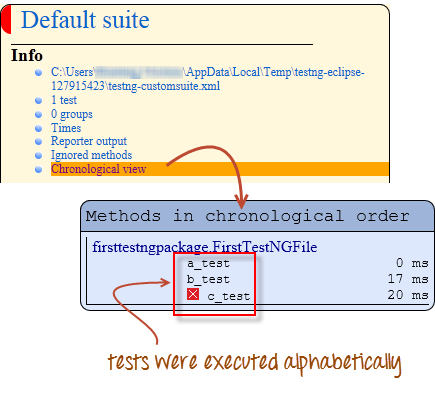
In the previous section, you have been introduced to the @Test annotation. Now, we shall be studying more advanced annotations and their usages.

**Multiple Test Cases**

We can use multiple @Test annotations in a single TestNG file. By default, methods annotated by @Test are executed alphabetically. See the code below. Though the methods c\_test, a\_test, and b\_test are not arranged alphabetically in the code, they will be executed as such.



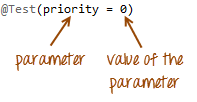
Run this code and on the generated index.html page, click “Chronological view.”



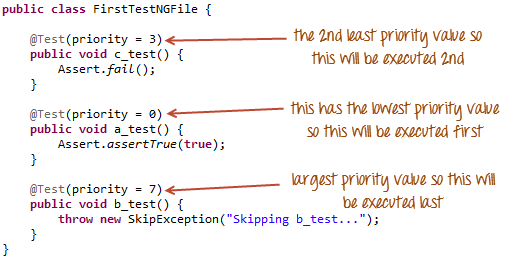
**Parameters**

If you want the methods to be executed in a different order, use the parameter “priority”. **Parameters are keywords that modify the annotation’s function**.

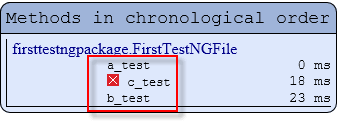
* Parameters require you to assign a value to them. You do.this by placing a “=” next to them, and then followed by the value.
* Parameters are enclosed in a pair of parentheses which are placed right after the annotation like the code snippet shown below.



TestNG will execute the @Test annotation with the lowest priority value up to the largest. There is no need for your priority values to be consecutive.



The TestNG HTML report will confirm that the methods were executed based on the ascending value of priority.



**Multiple Parameters**

Aside from “priority,” @Test has another parameter called “alwaysRun” which can only be set to either “true” or “false.” **To use two or more parameters in a single annotation, separate them with a comma** such as the one shown below.

@Test(priority = 0, alwaysRun = true)

@BeforeTest and @AfterTest

|  |  |
| --- | --- |
| **@BeforeTest** | methods under this annotation will be executed **prior to the first test case in the TestNG file**. |
| **@AfterTest** | methods under this annotation will be executed **after all test cases in the TestNG file are executed**. |

Consider the code below.

package firsttestngpackage;

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.\*;

public class firsttestngfile {

public String baseUrl = "http://demo.guru99.com/test/newtours/";

String driverPath = "C:\\geckodriver.exe";

public WebDriver driver ;

@BeforeTest

public void launchBrowser() {

System.out.println("launching firefox browser");

System.setProperty("webdriver.gecko.driver", driverPath);

driver = new FirefoxDriver();

driver.get(baseUrl);

}

@Test

public void verifyHomepageTitle() {

String expectedTitle = "Welcome: Mercury Tours";

String actualTitle = driver.getTitle();

Assert.assertEquals(actualTitle, expectedTitle);

}

@AfterTest

public void terminateBrowser(){

driver.close();

}

}

Applying the logic presented by the table and the code above, we can predict that the sequence by which methods will be executed is:

* 1st – launchBrowser()
* 2nd – verifyHomepageTitle()
* 3rd – terminateBrowser()

**The placement of the annotation blocks can be interchanged without affecting the chronological order by which they will be executed**. Let’s understand with a TestNG Example and try to rearrange the annotation blocks such that your code would look similar to the one below.

package firsttestngpackage;

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.\*;

public class firsttestngfile {

public String baseUrl = "http://demo.guru99.com/test/newtours/";

String driverPath = "C:\\geckodriver.exe";

public WebDriver driver ;

@AfterTest //Jumbled

public void terminateBrowser(){

driver.close();

}

@BeforeTest //Jumbled

public void launchBrowser() {

System.out.println("launching firefox browser");

System.setProperty("webdriver.gecko.driver", driverPath);

driver = new FirefoxDriver();

driver.get(baseUrl);

}

@Test //Jumbled

public void verifyHomepageTitle() {

String expectedTitle = "Welcome: Mercury Tours";

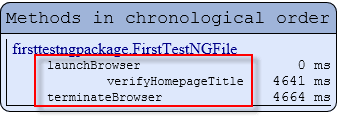
String actualTitle = driver.getTitle();

Assert.assertEquals(actualTitle, expectedTitle);

}

}

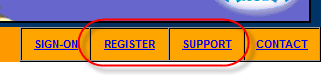
Run the code above and notice that



@BeforeMethod and @AfterMethod

|  |  |
| --- | --- |
| **@BeforeMethod** | methods under this annotation will be executed **prior to each method in each test case**. |
| **@AfterMethod** | methods under this annotation will be executed **after each method in each test case.** |

In Mercury Tours, suppose we like to verify the titles of the target pages of the two links below.



The flow of our test would be:

* Go to the homepage and verify its title.
* Click REGISTER and verify the title of its target page.
* Go back to the homepage and verify if it still has the correct title.
* Click SUPPORT and verify the title of its target page.
* Go back to the homepage and verify if it still has the correct title.

The code below illustrates how @BeforeMethod and @AfterMethod are used to efficiently execute the scenario mentioned above.

package firsttestngpackage;

import org.openqa.selenium.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.\*;

@Test

public class firsttestngfile {

public String baseUrl = "http://demo.guru99.com/test/newtours/";

String driverPath = "C:\\geckodriver.exe";

public WebDriver driver;

public String expected = null;

public String actual = null;

@BeforeTest

public void launchBrowser() {

System.out.println("launching firefox browser");

System.setProperty("webdriver.gecko.driver", driverPath);

driver= new FirefoxDriver();

driver.get(baseUrl);

}

@BeforeMethod

public void verifyHomepageTitle() {

String expectedTitle = "Welcome: Mercury Tours";

String actualTitle = driver.getTitle();

Assert.assertEquals(actualTitle, expectedTitle);

}

@Test(priority = 0)

public void register(){

driver.findElement(By.linkText("REGISTER")).click() ;

expected = "Register: Mercury Tours";

actual = driver.getTitle();

Assert.assertEquals(actual, expected);

}

@Test(priority = 1)

public void support() {

driver.findElement(By.linkText("SUPPORT")).click() ;

expected = "Under Construction: Mercury Tours";

actual = driver.getTitle();

Assert.assertEquals(actual, expected);

}

@AfterMethod

public void goBackToHomepage ( ) {

driver.findElement(By.linkText("Home")).click() ;

}

@AfterTest

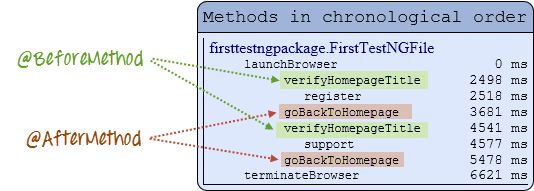
public void terminateBrowser(){

driver.close();

}

}

After executing this test, your TestNG should report the following sequence.



Simply put, @BeforeMethod should contain methods that you need to run **before** each test case while @AfterMethod should contain methods that you need to run **after** each test case.