**TestNG Groups with Example**

We use groups in [Testng](https://www.guru99.com/all-about-testng-and-selenium.html) when,

* We don't want to define test methods separately in different classes (depending upon functionality) and
* At the same time want to ignore (not to execute) some test cases as if they does not exist in the code.
* So to carry out this we have to Group them. This is done by using "include" and "exclude" mechanism supported in testNG.

In below example, we have shown the syntax of how to use groups in the XML file.

@Test (groups = { "bonding", "strong\_ties" })

Here we are using 2 group names i.e. "bonding" and "strong\_ties" (these are logical name that can be altered as per your wish).

**<groups> tag** defines the starting of groups in XML.

Customize your XML to pick the mentioned group from the test classes. Below mentioned is the syntax of how to declare groups in XML file e.g.

<groups>

<run>

<include name="bonding" />

</run>

</groups>

So, let us assume that there are 10 test methods in a class.

Out of them,

* 6 methods are tagged in "bonding" group and
* 4 are in "strong\_ties" group

Moving forward, we are going to set maven/Java path and use the Eclipse IDE to demonstrate the usage of groups using XML files in [Java](https://www.guru99.com/java-tutorial.html) based maven project.

**Set maven and Java path in environment variable (for windows user)**

Please refer <https://www.guru99.com/maven-jenkins-with-selenium-complete-tutorial.html>

<https://www.guru99.com/install-java.html>

**Introduction to XML and how to make an XML files**

* XML (Extensible Markup Language) file in Maven framework contains the information of one or more tests and is defined by the **tag <suite>**.
* Test information in XML is represented by **tag <test>** and can contain one or more TestNG classes.
* A Java class which contains **@Test** annotation above test methods is defined as TestNG methods.

Multiple tags are used in a sequence to build a working testNG xml like <suite>, <test> and <class>

* First is <suite> tag, which holds a logical name which defines full information to testNG reported to generate execution report.
* Second is <test name="Guru 99 Smoke Test Demo">, note it is logical name which holds the information of test execution report like pass, fail, skip test cases and other information like total time for execution and group info
* Third is <class name="**com.group.guru99.TC\_Class1**" />, com.group.guru99 is the package used, and Test Class name is TC\_Class1.

<?xml version="1.0" encoding="UTF-8" ?>

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

<suite name="**Suite**">

<test name="**Guru 99 Smoke Test Demo**">

<groups>

<run>

<include name="strong\_ties" />

</run>

</groups>

<classes>

<class name="**com.group.guru99.TC\_Class1**" />

</classes>

</test>

</suite>

We will be using this XML for upcoming video downside.

**Another mechanism instead of Grouping is "exclude" or "include" in test XML**

Suppose you are finding the usage of group mechanism complex then testNG XML facilitate the functionality to exclude/include a test.

**Exclude Tag**: Syntax for exclude tag <exclude name="${TEST\_CASE\_NAME}" />

**Include Tag**: Syntax for include tag <include name="${TEST\_CASE\_NAME}" />

**Note:** We can include/exclude multiple test cases once at a time, and it works with Groups as well.

package com.group.guru99;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.Test;

public class TC\_Class1 {

public static final WebDriver webDriver = new FirefoxDriver();;

String launchPageHeading = "//h3[text()='Guru99 Bank']";

final String userName\_element = "//input[@name='uid']", password\_element = "//input[@name='password']",

signIn\_element = "//input[@name='btnLogin']";

final String userName\_value = "mngr28642", password\_value = "ydAnate";

final String managerID = "//td[contains(text(),'Manger Id')]";

final String newCustomer = "//a[@href='addcustomerpage.php']", fundTransfer = "//a[@href='FundTransInput.php']";

/\*\*

\* This test case will initialize the webDriver

\*/

@Test(groups = { "bonding", "strong\_ties" })

public void tc01LaunchURL() {

webDriver.manage().window().maximize();

webDriver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

webDriver.get("http://www.demo.guru99.com/V4/");

}

/\*\*

\* Will check the presence of Heading on Login Page

\*/

@Test(groups = { "bonding" })

public void tc02VerifyLaunchPage() {

Assert.assertTrue(webDriver.findElement(By.xpath(launchPageHeading)).isDisplayed(),

"Home Page heading is not displayed");

System.out.println("Home Page heading is displayed");

}

/\*\*

\* This test case will enter User name, password and will then click on

\* signIn button

\*/

@Test(groups = { "bonding", "strong\_ties" })

public void tc03EnterCredentials() {

webDriver.findElement(By.xpath(userName\_element)).sendKeys(userName\_value);

webDriver.findElement(By.xpath(password\_element)).sendKeys(password\_value);

webDriver.findElement(By.xpath(signIn\_element)).click();

}

/\*\*

\* This test case will verify manger's ID presence on DashBoard

\*/

@Test(groups = { "strong\_ties" })

public void tc04VerifyLoggedInPage() {

Assert.assertTrue(webDriver.findElement(By.xpath(managerID)).isDisplayed(),

"Manager ID label is not displayed");

System.out.println("Manger Id label is displayed");

}

/\*\*

\* This test case will check the presence of presence of New customer link

\* And FundTransfer link in Left pannel

\*/

@Test(groups = { "bonding" })

public void tc05VerifyHyperlinks() {

Assert.assertTrue(webDriver.findElement(By.xpath(newCustomer)).isEnabled(),

"New customer hyperlink is not displayed");

System.out.println("New customer hyperlink is displayed");

Assert.assertTrue(webDriver.findElement(By.xpath(fundTransfer)).isEnabled(),

"Fund Transfer hyperlink is not displayed");

System.out.println("Fund Transfer hyperlink is displayed");

}

}

**Explanation of Code:**

As mentioned above, we have created 5 test cases for performing each action in independent methods.

You can observe that to every method, we have associated a group parameter holding some value in it.

Basically, these are the name of the differentiating groups i.e. "strong\_ties" & "bonding".

* First and Third methods are tagged to "bonding", "strong\_ties" which means if XML is updated in any of the group, this [Test Case](https://www.guru99.com/test-case.html) will run.
* The second method is only tagged to "bonding" group it means that if XML is updated with bonding group. Only in that case this test case will run.
* Fourth Test case is tagged to strong\_ties group, which means this test case will only run if XML is updated with strong\_ties group name.
* Last but not the least fifth test case is attached to bonding group, which means this test case will only run if XML is updated with bonding group name.