

# Chapter 7.3:

## Groups & Depends On Groups

### Introduction

In this chapter, we will discuss the Groups Attribute, Depends On Groups Attribute, and Execute Groups At Runtime using an xml File.

### Groups Attribute

The Groups Attribute is a way to add any number of Test Methods to a named list. This feature is beneficial because it's an opportunity to run only those lists of Test Methods according to their group name. The group names can be a module of our application, a certain Test Type, whatever we decide. Plus, one Test Method can be a part of more than one group. For example, you can add the Sign In Test Method to a group for Smoke Test and to another group for Regression Test. Smoke Test is a Test Type that makes sure the most important functions in an application still work after a build is deployed to an environment. Regression is when a set of Test Scripts are re-executed to make sure the existing functions are still working after an application change.

In this example, test1\_SetUpChrome is added to a group called initialize. The description for groups show "The list of groups this class/method belongs to". A class can also be added to a group.

### Depends On Groups Attribute

The Depends On Groups Attribute is similar to the Depends On Methods attribute. Depends On Groups allow us to define a group name that our Test Method depends on. The next Test Method test2\_OpenOrangeHRM depends on the initialize group. Hover over the attribute. The description states "The list of groups this method depends on. Every method member of one of these groups is guaranteed to have been invoked before this method."

Also, this Test Method is part of a group called env\_application. Let's pretend there is a different URL for each environment such as DEV Environment, QA Environment, and Production Environment. So, we have a group name for those environments.

We are not forced to add our Test Methods to a group. The remaining Test Methods do not belong to a group but only depends on a group. They depend on group env\_application: test3\_SignIn, test4\_SearchUser, test5\_SearchEmployee, test6\_SearchCandidate, and test7\_SignOut all depend on the same group.

Let's run. We see the application is in Spanish. It was changed. All Test Methods Pass. The Results tab also shows green for all Test Methods. Now, let's run and see what happens when there's a Failure. Could not connect to the application. There's 1 Failure and 5 Skips just like with the Depends On Methods attribute.

## Execute Groups At Runtime via xml File

Execute Groups At Runtime using xml. TestNG provides a few ways to run groups at Runtime. The main way is to use an xml file but we can also use the command line or ant. Here's a fake Amazon class site with 6 Test Methods. Test 1 is Login and has been added to the smoke test group. Test 2 is Search Products and has been added to the smoke test and regression test groups. Test 3 is Place Order and it has been added to the regression test, integration test, and defect group. We have the option of being very descriptive with our group name so I added dot fix. This Test Method had a defect that should be fixed. If it is fixed then we can remove Place Order from the defect group.

Why do we have a defect group? We can add a defect group for a few reasons. When a Test Method keeps failing, we can add it to the defect group to avoid running that test because we know it's not fixed yet. On the flip side, we can execute only the defect group then verify which Test Methods are still broken.

Test 4 Send Confirmation has been added to system testing and defect.backlog group. This defect will not get fixed no time soon. Test 5 is Ship Order and has been added to the groups regression and defect.progress. Test 6 is Log Out and it has no groups.

Let's execute some scenarios for groups at runtime using the xml file – All Groups. Notice, the groups tag and run tags. The groups tag allows 2 tags. We see the run tag which list the groups that can be included or excluded. This file does not include or exclude any groups. Therefore, all groups will show up. The groups tag also allows a define tag which help us create a new group based on existing groups. Let's Run the Test Suite and we see all groups. Test Methods 1 – 6 show up and Passed.

The next xml file includes a regression group. The Console show Test Methods added to the regression group. We see other groups because those Test Methods are part of more than 1 group.

In the xml file, we can include more than 1 group at Runtime. This file includes the smoke test and system test group. There are 3 Test Methods.

Look what happens when we exclude regression. As expected, there are no Test Methods in the Console assigned to a regression group.

We have the ability to use regular expressions. In this xml file, defects have a regular expression using a dot and asterisk. Out all of those Test Methods assigned to regression. Only 1 Test Method shows up. That's because, the other Test Methods are assigned to a regression group and defect group. Therefore, it's not in the Console because defects are not included.

## Run Test Methods According To Group Name

Let's recap the scenarios we covered when executing groups at Runtime. A run that has No Include and No Exclude will run all Test Methods. If we include a group like regression then only Test Methods assigned to that group will run. The same goes if we include more than 1 group. We used smoke and system. Only Test Methods assigned to those groups were executed. Excluding a group name runs all Test Methods except the Test Methods assigned to that group. Last but not least, we can use regular expressions. In this example, defect used a regular expression. If a Test Method belongs to a group that

is included and excluded then it will not run because excluded wins. There were 3 Test Methods assigned to regression but only 1 showed up. The other 2 did not show up because they were assigned to regression and defect.

So far, up to this point, all of the Test Methods were executed from the same class. However, TestNG is not limited to executing a group using 1 class. We can execute groups using more than 1 class or even a package. This xml file has 2 class tags. By the way, we write the class name using the package name, dot operator sub package name, dot operator then the class name. For example, the first class name shows com.testautomationu. That's the package name and sub package name is chp6dependencyteting then we see Amazon\_FakeSite. For this class, we include the regression group.

For the last class, we include env\_application. Recall this class, include test2\_OpenOrangeHRM that is assigned to group env\_application.

Let's Run. All of the Test Methods show up. We saw the Test Script open OrangeHRM. The Results tab reflect the Console by showing both classes.

Next, we will discuss Data Driven Testing which allow us to execute 1 Test Script with more than 1 set of Test Data.