**Using DataProvider in TestNG to make the tests individual:**

We have seen data provider in earlier videos. In our current project we are reading the main test cases file in the method startExecution() (earlier it is main method) and passing the information test case no, tests work book name and sheet name to the method prepareKeywords().

Now we use dataprovider annotation here and make this data as multidimensional array so that TestNG considers each test as separate test case. For this we need to modify startExecution() method.

startExecution() method:

@DataProvider(name = "maintestcases")

**public** **static** String[][] startexecution() **throws** Exception{

excelUtilities eu = **new** excelUtilities();

Properties gldata = **new** Properties();

InputStream input = **new** FileInputStream("src/executionEngine/config.properties");

gldata.load(input);

List<List<String>> testcases = **new** ArrayList<List<String>>();

testcases = eu.getTestCases(gldata.getProperty("WB\_PATH\_TESTS"), gldata.getProperty("WB\_PATH\_TESTS\_SHEET"));

**int** no\_test\_cases = testcases.size();

String testcasesobject[][] = **new** String[no\_test\_cases][3];

**for**(**int** i=0; i<testcases.size(); i++) {

testcasesobject[i][0] = testcases.get(i).get(0);

testcasesobject[i][1] = testcases.get(i).get(1);

testcasesobject[i][2] = testcases.get(i).get(2);

}

System.***out***.println(Arrays.*toString*(testcasesobject));

**return** testcasesobject;

}

Here we provided the DataProvider annotation and a name to the dataprovider. Here we are doing the same procedure as earlier but we are reading the information into a multidimensional array testcasesobject and we are returning this.

Now we have to get this data into the prepareKeywords() method and use it there. So we need to change the prepareKeywords() method parameters. Here the dataprovider is providing three values – test case name, workbook name and sheet name. So we will change the prepareKeywords() parameters accordingly.

prepareKeywords(String testcase, String wbbook, String sheet)

The following is the updated prepareKeywrods() method.

@Test(dataProvider = "maintestcases")

**public** **static** **void** prepareKeywords(String testcase, String wbbook, String sheet) **throws** IOException, ClassNotFoundException, NoSuchMethodException, SecurityException, IllegalAccessException, IllegalArgumentException, InvocationTargetException, InstantiationException, InterruptedException {

Properties gldata = **new** Properties();

InputStream input = **new** FileInputStream("src/executionEngine/config.properties");

gldata.load(input);

excelUtilities ecu = **new** excelUtilities();

List<String> keywords = **new** ArrayList<String>();

keywords = ecu.getKeywordsFromTestCase(testcase,gldata.getProperty("WB\_PATH\_TEST\_CASES")+wbbook+".xlsx", sheet);

//System.out.println(keywords);

*getDependencies*(keywords);

}

We have used @Test for this method and same dataProvider name as that of startExecution method. This means that, this method will receive data from startExecution method. Now TestNG considers each method as a parallel one. Now if you run the test cases you will see that TestNG considers each test as a separate one and it won’t stop execution even if one test case fails.

The following are the results.

===============================================

Suite

Total tests run: 43, Failures: 22, Skips: 0

===============================================

You can see that now TestNG is considering each test case as separate one.

But if any test fails, the browser won’t be closed. In the above case around 22 failures were there so 22 firefox browsers are in open state. This will make the execution slower. So we have to close the browser when any test fails. We will do it using TestNG listeners.