**Dataprovider & TestNG XML: Parameterization in Selenium(Example)**

As we create software, we always wish it should work differently with a different set of data. When it comes to[Testing](https://www.guru99.com/software-testing.html)the same piece of software, we can’t be unfair to test it with just one set of data. Here again, we need to verify that our system is taking all set of combinations which are expected to support. For that, we need to parameterize our test scipts. Here comes Parameterization in the picture.

**Parameterization in Selenium**

**Parameterization in Selenium** is a process to parameterize the test scripts in order to pass multiple data to the application at runtime. It is a strategy of execution which automatically runs test cases multiple times using different values. The concept achieved by parameterizing the test scripts is called **Data Driven Testing**.

**Type of Parameterization in TestNG-**

To make parameterization more clear, we will go through the parameterization options in one the most popular framework for Selenium Webdriver – **TestNG**.

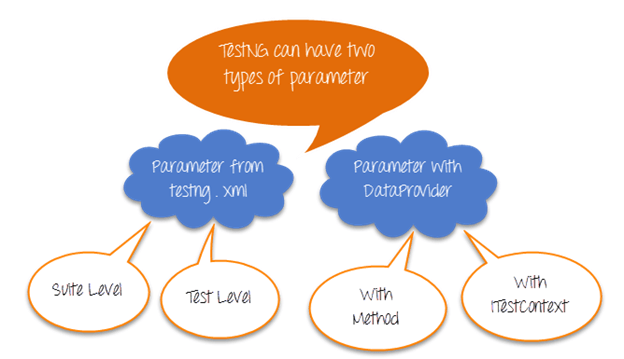
There are **two ways** by which we can achieve parameterization in TestNG

1. With the help of **Parameters** **annotation** and **TestNG XML** file.

TestNG: Parameterization using XML & DataProvider in Selenium

1. With the help of **DataProvider** annotation.

TestNG: Parameterization using XML & DataProvider in Selenium



Parameters from Testng.xml can be suite or test level

**Parameters Annotation in TestNG**

**Parameters Annotation in TestNG** is a method used to pass values to the test methods as arguments using .xml file. Users may be required to pass the values to the test methods during run time. The @Parameters annotation method can be used in any method having @Test, @Before, @After or @Factory annotation.

**Parameters annotation with Testng.xml**

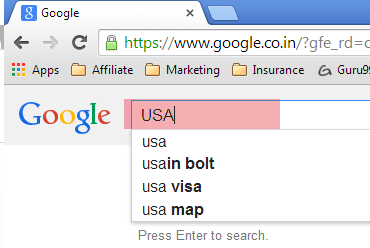
Select parameterization using annotations when you do want to deal with complexity & the number of input combinations are less.

Let see how this works

Test Scenario

Step 1) Launch browser & go to Google.com

Step 2) Enter a search keyword



Step 3) Verify the inputted value is same as that provided by our test data

Step 4) Repeat 2 & 3 until all values are inputted

|  |  |
| --- | --- |
| **Test Author** | **SearchKey** |
| Guru99 | India |
| Krishna | USA |
| Bhupesh | China |

Here is an example of how to do it WITHOUT parameters

package parameters;

import org.testng.annotations.Test;

import org.testng.AssertJUnit;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

public class NoParameterWithTestNGXML {

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

WebDriver driver;

@Test

public void testNoParameter() throws InterruptedException{

String author = "guru99";

String searchKey = "india";

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

driver= new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("https://google.com");

WebElement searchText = driver.findElement(By.name("q"));

//Searching text in google text box

searchText.sendKeys(searchKey);

System.out.println("Welcome ->"+author+" Your search key is->"+searchKey);

System.out.println("Thread will sleep now");

Thread.sleep(3000);

System.out.println("Value in Google Search Box = "+searchText.getAttribute("value") +" ::: Value given by input = "+searchKey);

//verifying the value in google search box

AssertJUnit.assertTrue(searchText.getAttribute("value").equalsIgnoreCase(searchKey));

}

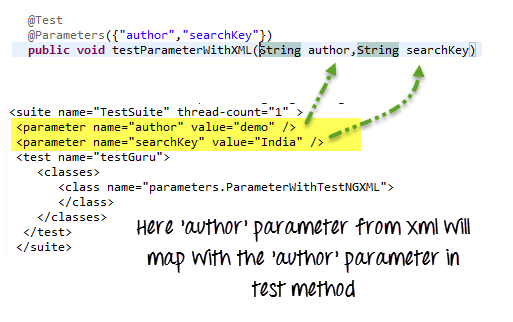
}

A Study, the above example. Just imagine how complex the code will become when we do this for 3 input combinations

Now, let’s parameterize this using TestNG

To do so, you will need to

* Create an XML file which will store the parameters
* In the test, add annotation @Parameters



Here is the complete code

**Test Level TestNG.xml**

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

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

<suite name="TestSuite" thread-count="3" >

<parameter name="author" value="Guru99" />

<parameter name="searchKey" value="India" />

<test name="testGuru">

<parameter name="searchKey" value="UK" />

<classes>

<class name="parameters.ParameterWithTestNGXML">

</class>

</classes>

</test>

</suite>

**ParameterWithTestNGXML.java File**

package parameters;

import org.testng.AssertJUnit;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.Optional;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

public class ParameterWithTestNGXML {

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

WebDriver driver;

@Test

@Parameters({"author","searchKey"})

public void testParameterWithXML( @Optional("Abc") String author,String searchKey) throws InterruptedException{

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

driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("https://google.com");

WebElement searchText = driver.findElement(By.name("q"));

//Searching text in google text box

searchText.sendKeys(searchKey);

System.out.println("Welcome ->"+author+" Your search key is->"+searchKey);

System.out.println("Thread will sleep now");

Thread.sleep(3000);

System.out.println("Value in Google Search Box = "+searchText.getAttribute("value") +" ::: Value given by input = "+searchKey);

//verifying the value in google search box

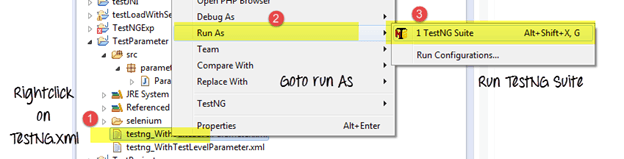
AssertJUnit.assertTrue(searchText.getAttribute("value").equalsIgnoreCase(searchKey));

}

}

Instructions to run the script, select the XML file and Run as Test NG Suite

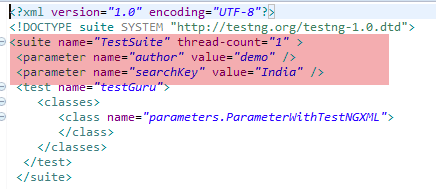
**Right Click on .xml file -> Run as ->**[**Testng**](https://www.guru99.com/all-about-testng-and-selenium.html)**Suite (Note : Suite)**



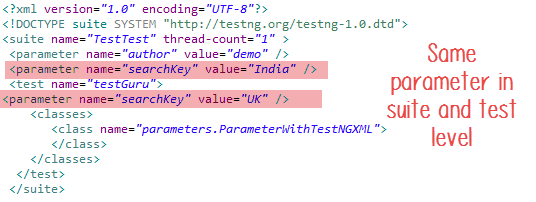
Now, parameters can be defined at 2 levels

1. Suite level – The parameters inside the <suite> tag of TestNG XML file will be a suite level parameter.
2. Test Level — The parameters inside the <Test> tag of testing XML file will be a Test level parameter.

Here is the same test with suite level parameters



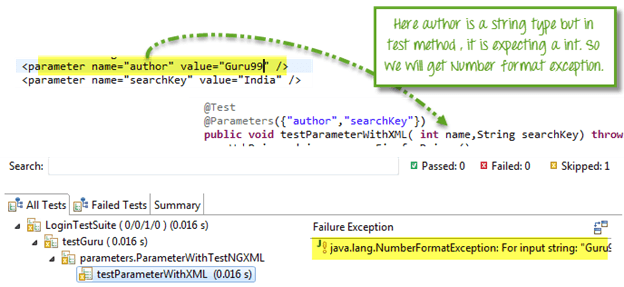
***NOTE:***In case if the parameter name is same in suite level and test level then test level parameter will get preference over suite level. So, in that case, all the classes inside that test level will share the overridden parameter, and other classes which are outside the test level will share suite level parameter.



**Troubleshooting**

***Issue # 1***Parameter value in testng.xml cannot be typecasted to the corresponding test method’s parameter it will throw an error.

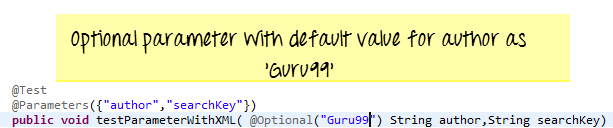
Consider the following example



Here, ‘author’ attribute is equal to ‘Guru99’ which is a string and in corresponding test method its expecting an integer value, so we will get an exception here.

***Issue # 2***Your @Parameters do not have a corresponding value in testing.xml.

You can solve this situation by adding **@optional** **annotation** in the corresponding parameter in the test method.



***Issue # 3*:**You want to test multiple values of the same parameter using Testng.xml

The Simple answer is this can not be done! You can have multiple different parameters, but each parameter can only have a single value. This helps prevent hardcoding values into the script. This makes code reusable. Think of it as config files for your script. If you want to use multiple values for a parameter use DataProviders

**Data Provider in TestNG**

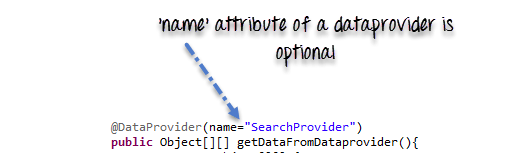
**Data Provider in TestNG** is a method used when a user needs to pass complex parameters. Complex Parameters need to be created from Java such as complex objects, objects from property files or from a database can be passed by the data provider method. The method is annotated by @DataProvider and it returns an array of objects.

**Parameters using Dataprovider**

@Parameters annotation is easy but to test with multiple sets of data we need to use Data Provider.

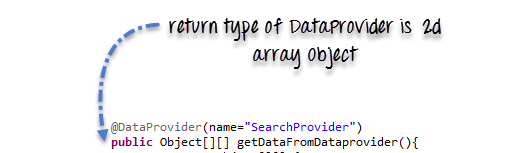
To fill thousand’s of web forms using our testing framework we need a different methodology which can give us a very large dataset in a single execution flow.

This data driven concept is achieved by **@DataProvider**annotation in TestNG.

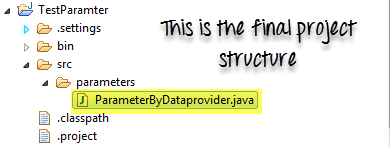


It has only one **attribute ‘name’**. If you do not specify the name attribute then the DataProvider’s name will be same as the corresponding method name.

Data provider returns **a two-dimensional JAVA object** to the test method and the test method, will invoke M times in a M\*N type of object array. For example, if the DataProvider returns an array of 2\*3 objects, the corresponding testcase will be invoked 2 times with 3 parameters each time.



**Complete Example**



package parameters;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

public class ParameterByDataprovider {

WebDriver driver;

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

@BeforeTest

public void setup(){

//Create firefox driver object

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

driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("https://google.com");

}

/\*\* Test case to verify google search box

\* @param author

\* @param searchKey

\* @throws InterruptedException

\*/

@Test(dataProvider="SearchProvider")

public void testMethod(String author,String searchKey) throws InterruptedException{

{

WebElement searchText = driver.findElement(By.name("q"));

//search value in google searchbox

searchText.sendKeys(searchKey);

System.out.println("Welcome ->"+author+" Your search key is->"+searchKey);

Thread.sleep(3000);

String testValue = searchText.getAttribute("value");

System.out.println(testValue +"::::"+searchKey);

searchText.clear();

//Verify if the value in google search box is correct

Assert.assertTrue(testValue.equalsIgnoreCase(searchKey));

}

}

/\*\*

\* @return Object[][] where first column contains 'author'

\* and second column contains 'searchKey'

\*/

@DataProvider(name="SearchProvider")

public Object[][] getDataFromDataprovider(){

return new Object[][]

{

{ "Guru99", "India" },

{ "Krishna", "UK" },

{ "Bhupesh", "USA" }

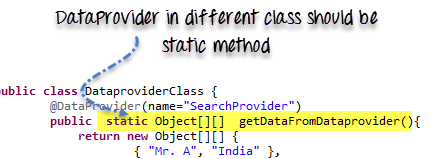
};

}

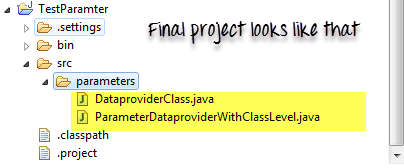
}

**Invoke DataProvider from different class**

By default, DataProvider resides in the same class where test method is or its base class. To put it in some other class we need to make data provider method as static and in test method we need to add an attribute **dataProviderClass** in**@Test** annotation.



**Code Example**



**TestClass ParameterDataproviderWithClassLevel.java**

package parameters;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class ParameterDataproviderWithClassLevel {

WebDriver driver;

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

@BeforeTest

public void setup(){

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

driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("https://google.com");

}

@Test(dataProvider="SearchProvider",dataProviderClass=DataproviderClass.class)

public void testMethod(String author,String searchKey) throws InterruptedException{

WebElement searchText = driver.findElement(By.name("q"));

//Search text in google text box

searchText.sendKeys(searchKey);

System.out.println("Welcome ->"+author+" Your search key is->"+searchKey);

Thread.sleep(3000);

//get text from search box

String testValue = searchText.getAttribute("value");

System.out.println(testValue +"::::"+searchKey);

searchText.clear();

//verify if search box has correct value

Assert.assertTrue(testValue.equalsIgnoreCase(searchKey));

}

}

**DataproviderClass.java**

package parameters;

import org.testng.annotations.DataProvider;

public class DataproviderClass {

@DataProvider(name="SearchProvider")

public static Object[][] getDataFromDataprovider(){

return new Object[][] {

{ "Guru99", "India" },

{ "Krishna", "UK" },

{ "Bhupesh", "USA" }

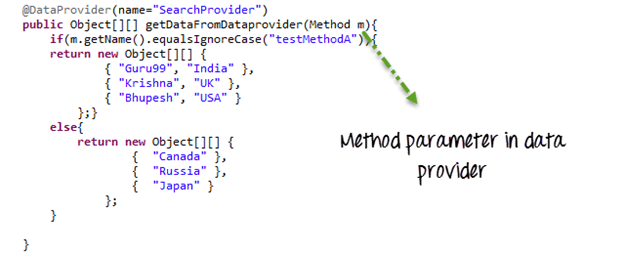
};

}}

**Types of Parameters in Dataprovider**

There are two type of parameters supported by DataProvider method.

**Method**– If the **SAME** DataProvider should behave differently with different test method , use Method parameter.



In the following example ,

* We check if method name is testMethodA.
* If yes return one set of value
* Else return another set of value

package parameters;

import java.lang.reflect.Method;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

public class ParameterByMethodInDataprovider{

WebDriver driver;

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

@BeforeTest

public void setup(){

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

driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("https://google.com");

}

@Test(dataProvider="SearchProvider")

public void testMethodA(String author,String searchKey) throws InterruptedException{

WebElement searchText = driver.findElement(By.name("q"));

//Search text in search box

searchText.sendKeys(searchKey);

//Print author and search string

System.out.println("Welcome ->"+author+" Your search key is->"+searchKey);

Thread.sleep(3000);

String testValue = searchText.getAttribute("value");

System.out.println(testValue +"::::"+searchKey);

searchText.clear();

//Verify if google text box is showing correct value

Assert.assertTrue(testValue.equalsIgnoreCase(searchKey));

}

@Test(dataProvider="SearchProvider")

public void testMethodB(String searchKey) throws InterruptedException{

{

WebElement searchText = driver.findElement(By.name("q"));

//Search text in search box

searchText.sendKeys(searchKey);

//Print only search string

System.out.println("Welcome ->Unknown user Your search key is->"+searchKey);

Thread.sleep(3000);

String testValue = searchText.getAttribute("value");

System.out.println(testValue +"::::"+searchKey);

searchText.clear();

//Verify if google text box is showing correct value

Assert.assertTrue(testValue.equalsIgnoreCase(searchKey));

}

}

/\*\*

\* Here DataProvider returning value on the basis of test method name

\* @param m

\* @return

\*\*/

@DataProvider(name="SearchProvider")

public Object[][] getDataFromDataprovider(Method m){

if(m.getName().equalsIgnoreCase("testMethodA")){

return new Object[][] {

{ "Guru99", "India" },

{ "Krishna", "UK" },

{ "Bhupesh", "USA" }

};}

else{

return new Object[][] {

{ "Canada" },

{ "Russia" },

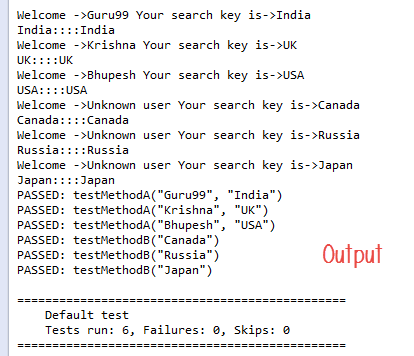
{ "Japan" }

};}

}

}

Here is the output



**ITestContext**– It can use to create different parameters for test cases based on groups.

In real-life, you can use ITestContext to vary parameter values based on Test Methods, hosts, configurations of the test.



In the following code example

* We have 2 groups A & B
* Each test method is assigned to a group
* If value of group is A, a particular data set is returned
* If value of group is B, another data set is returned

package parameters;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.ITestContext;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

public class ParameterByITestContextInDataprovider {

WebDriver driver;

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

@BeforeTest(groups={"A","B"})

public void setup(){

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

driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("https://google.com");

}

@Test(dataProvider="SearchProvider",groups="A")

public void testMethodA(String author,String searchKey) throws InterruptedException{

{

//search google textbox

WebElement searchText = driver.findElement(By.name("q"));

//search a value on it

searchText.sendKeys(searchKey);

System.out.println("Welcome ->"+author+" Your search key is->"+searchKey);

Thread.sleep(3000);

String testValue = searchText.getAttribute("value");

System.out.println(testValue +"::::"+searchKey);

searchText.clear();

//verify correct value in searchbox

Assert.assertTrue(testValue.equalsIgnoreCase(searchKey));

}

}

@Test(dataProvider="SearchProvider",groups="B")

public void testMethodB(String searchKey) throws InterruptedException{

{

//find google search box

WebElement searchText = driver.findElement(By.name("q"));

//search a value on it

searchText.sendKeys(searchKey);

System.out.println("Welcome ->Unknown user Your search key is->"+searchKey);

Thread.sleep(3000);

String testValue = searchText.getAttribute("value");

System.out.println(testValue +"::::"+searchKey);

searchText.clear();

//verify correct value in searchbox

Assert.assertTrue(testValue.equalsIgnoreCase(searchKey));

}

}

/\*\*

\* Here the DAtaProvider will provide Object array on the basis on ITestContext

\* @param c

\* @return

\*/

@DataProvider(name="SearchProvider")

public Object[][] getDataFromDataprovider(ITestContext c){

Object[][] groupArray = null;

for (String group : c.getIncludedGroups()) {

if(group.equalsIgnoreCase("A")){

groupArray = new Object[][] {

{ "Guru99", "India" },

{ "Krishna", "UK" },

{ "Bhupesh", "USA" }

};

break;

}

else if(group.equalsIgnoreCase("B"))

{

groupArray = new Object[][] {

{ "Canada" },

{ "Russia" },

{ "Japan" }

};

}

break;

}

return groupArray;

}

}

Note: If you directly run your testng class, it will first call dataprovider which can’t get groups information as groups are not available. But instead if you call this class via testng.xml, it will have groups info available with ITestContext. Use the following XML to call the test

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

<suite name="test-parameter">

<test name="example1">

<groups>

<run>

<include name="A" />

</run>

</groups>

<classes>

<class

name="parameters.ParameterByITestContextInDataprovider" />

</classes>

</test>

<test name="example2">

<groups>

<run>

<include name="B" />

</run>

</groups>

<classes>

<class

name="parameters.ParameterByITestContextInDataprovider" />

</classes>

</test>

</suite>

**Summary**

* **Parameterization** is require to create **Data Driven Testing**.
* TestNG support two kinds of parameterization, using **@Parameter+TestNG.xml** and using**@DataProvider**
* In**@Parameter+TestNG.xml**parameters can be placed in suite level and test level. If

The Same parameter name is declared in both places; test level parameter will get preference over suit level parameter.

* using @Parameter+TestNG.xml only one value can be set at a time, but @DataProvider return **an 2d array of Object**.
* If DataProvider is present in the different class then the class where the test method resides,**DataProvider** should be **static method**.
* There are two parameters supported by **DataProvider** are **Method** and **ITestContext.**