**Parallel Test Execution In TestNG With Selenium**

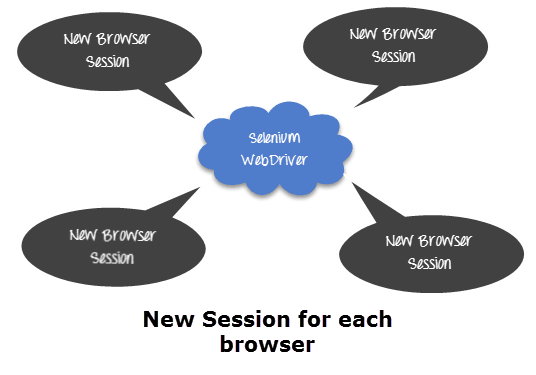
The evolution in the process of software delivery in organizations in response to business agility has resulted in a paradigm shift from traditional release cycles to continuous release models. To achieve the product delivery objectives in this new paradigm, [continuous testing](https://www.lambdatest.com/blog/continuous-testing-using-shift-left-testing-approach/) plays a vital role in ensuring the quality of end-to-end processes, along with ensuring effective collaboration between Quality Assurance (QA) and development teams.

The new era of continuous testing demands QA teams to explore solutions to [increase test execution speed](https://www.lambdatest.com/blog/speed-up-selenium-test-cases-execution/). Enterprises are looking to leverage the benefits offered by automation testing, while ensuring that manual testing is used in scenarios where automation is not possible.

The traditional approach of manual testing is already taken over by automation testing. One popular [automation testing strategy](https://www.lambdatest.com/blog/the-definitive-guide-to-automation-testing-for-it-teams/) widely adopted by QA teams to further reduce the testing time while ensuring maximum test coverage is the shift from sequential testing to [parallel testing](https://www.lambdatest.com/blog/what-is-parallel-testing-and-why-to-adopt-it/). You can perform parallel test execution in TestNG or any other framework of your choice with Selenium.

## Why do we need Session Handling?

During test execution, the Selenium WebDriver has to interact with the browser all the time to execute given commands. At the time of execution, it is also possible that, before current execution completes, someone else starts execution of another script, in the same machine and in the same type of browser.

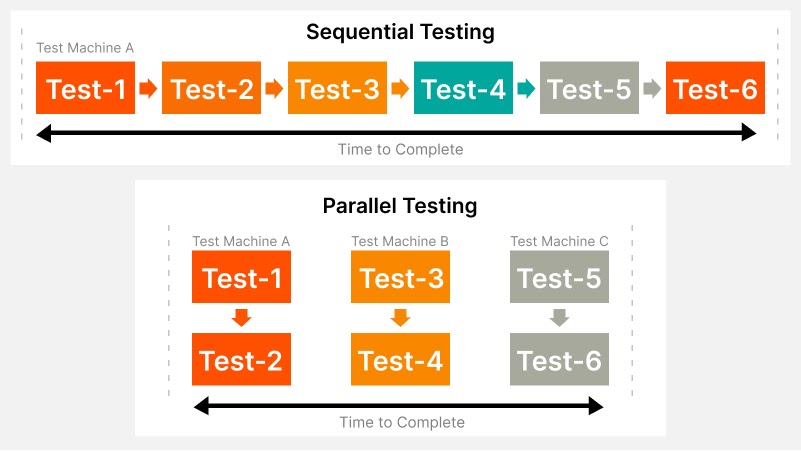


## Introduction to Parallel Test Execution in TestNG

Parallel testing is an approach in which independent tests are run parallel to reduce overall test execution time. A tester runs two or more versions of the software concurrently with the same input or test methods. Each test can be run on multiple devices, browsers, and OS combinations.

The parallel testing process can be applied to a single application, multiple applications, or even multiple components of an application. Parallel execution in TestNG can be achieved at different levels – threads, methods, etc.

Parallel testing was introduced to replace the traditional approach of sequential testing. As the name suggests, in sequential test execution tests are executed one after another, which is more time-consuming.



Because of the growing demand for cross browser testing in today’s market, parallel testing is implemented heavily with Selenium automation testing.

### **Advantages of Parallel Test Execution in TestNG**

Parallel testing is a powerful approach that allows you to reduce the time taken to perform [automation testing](https://www.lambdatest.com/learning-hub/automation-testing). There are various reasons that make it important for testers to adopt parallel testing in Selenium. Some of the most popular reasons are:

* **Accelerated Test Execution Speed**

If a sequential test takes 1 minute to complete a single test execution, it will take 5 minutes to run five tests one after another on the same device. However, overall test execution will take 1 minute (or less) when the same test is run in parallel on 5 devices (or input combinations).

* **Increased Test Coverage and Wider Compatibility**

Parallel testing enables you to test across more devices (desktop or mobile), browsers, and OS combinations at once. This extended test coverage will help in risk mitigation, reducing the chance of unnoticed bugs and offering wider compatibility, which will be instrumental in shipping a top-quality product.

* **Overcome the Testing Bottleneck**

Testing is often the bottleneck of DevOps. Tedious testing cycles can hold up the entire release process. But this bottleneck can be removed by accelerated test execution through parallel testing.

* **Fix Bugs Earlier**

Earlier the testing is completed, earlier the bugs can be unearthed and fixed. In addition, parallel testing is results-driven, since we can test more scenarios in less time, and there is more actionable data towards enhancements.

* **Flexibility**

You can revert to sequential testing when necessary.

### **Challenges of Parallel Test Execution in TestNG**

To achieve full-fledged parallel testing, as with any type of testing, you may face some challenges. The common challenges that you will run into are often related to:

* Scalability
* Infrastructure limitations. This includes:
  + Availability of secure, enterprise-grade lab that meet all the requirements
  + Recurring investments involved in the maintenance and upgradation of test infrastructure.
* ***Fails On Dependent Modules***: Parallel testing allows independent running of modules simultaneously. Due to this, we cannot go ahead with modules that are dependent on each other, and this occurs quite frequently while testing. So, either we run serially or dissolve independence, which takes extra time and effort.
* ***Knowledge Of Program Flow***: The tester should be well-versed with the flow of the program to create parallel testing modules. A slight interdependency can bring down the whole test case execution. The tester should also know which modules to run in multiple threads and which ones to run in the same threads etc.

The popular strategy recommended to overcome these challenges and move towards parallel testing is to use cloud-based testing providers. Through [Cloud testing](https://www.lambdatest.com/blog/cloud-testing-tutorial/), teams can test their apps on devices and browsers hosted on the cloud infrastructure. By utilizing a cloud-based testing infrastructure, you no longer need to maintain in-house infrastructure.

Also, the scalability issues associated with the local testing infrastructure for covering all major browsers, their different versions, and OS will be taken care of. Needless to say, it is more cost-efficient as well.

## Where can we apply Parallel Test execution in TestNG?

Before starting with the code, let's answer a genuine question, "***Where can we apply parallel testing in TestNG?***". TestNG has its rules too. Although it is reasonably evident that parallel testing must be used with the test case methods to run them in parallel TestNG offers three more areas where we can go ahead with parallel testing, combining these four areas, parallel testing accepts the following keywords (values) in TestNG:

* ***Methods***: This will run the parallel tests on all *@Test* methods in TestNG.
* ***Tests***: All the test cases present inside the <test> tag will run with this value.
* ***Classes***: All the test cases present inside the classes that exist in the XML will run in parallel.
* ***Instances***: This value will run all the test cases parallelly inside the same instance.

Alright, we are now all set to run our first test case parallelly in TestNG using Selenium. (Refer Cross-Browser testing In TestNG before moving forward)

## How to perform Parallel Test Execution in TestNG?

TestNG (where NG stands for ‘Next Generation’) is an open-source [Java-based test automation framework](https://www.lambdatest.com/blog/top-10-java-testing-frameworks/) that has gained huge popularity among developers and testers. The framework itself is inspired by JUnit, another [popular unit testing framework](https://www.lambdatest.com/blog/11-best-unit-testing-frameworks-for-selenium-automation/) in the Java ecosystem. However, it offers a variety of advanced and distinctive features, making it much more powerful than its peers. One such remarkable feature is the seamless support for Parallel Testing.

The TestNG [test automation](https://www.lambdatest.com/automation-testing) framework allows you to run tests in parallel or multithreaded mode by utilizing the Multi-Threading concept of Java. Multi-Threading is the process of executing multiple threads simultaneously, without any dependence on each other. Therefore, exceptions occurring on one thread won’t affect the others. In TestNG, we enable parallel testing by making required changes in the configuration file – TestNG XML file.

### **Parallel Test Execution in Selenium using Configuring TestNG XML in TestNG**

TestNG XML is the test suite configuration file in TestNG, which helps you organize your tests by defining test suites, [grouping test cases in TestNG](https://www.lambdatest.com/blog/grouping-test-cases-in-testng/), and running them under one XML file. IN a nutshell, it helps in customizing the execution of tests.

It also allows you to run a single test file over numerous combinations of parameters and values specified in it. First, let us go through the key features of the TestNG XML file, which will be useful for this tutorial.

* Provides seamless parallel test execution
* Supports [parameterization in TestNG](https://www.lambdatest.com/blog/parameterization-in-testng-dataprovider-and-testng-xml-examples/) using @Parameter annotation
* Allows grouping of test methods and group execution
* Allows executing test cases inside multiple Java class files
* [TestNG listeners](https://www.lambdatest.com/blog/testng-listeners-in-selenium-webdriver-with-examples/) automatically create logs

A simple TestNG XML file without any configuration for the parallel tests will look like this.

Graphical user interface, text, application

Description automatically generated

To trigger parallel test execution in TestNG, i.e., run tests on separate threads, we need to set the parallel attribute. This attribute accepts four values:

* **methods**– runs all methods with @Test annotation in parallel mode
* **tests** – runs all test cases present inside <test> tag in the XML in parallel mode
* **classes**– runs all test cases present inside classes in the XML in parallel mode
* **instances**– runs all test cases in the same instance in parallel mode

We can also set the number of threads we wish to allocate during the execution using the thread-count attribute. TestNG has a default value of thread = 5.

|  |  |
| --- | --- |
| 1 | <suite name="Regression Test Suite" parallel="methods" thread-count="2"> |

Here, we have defined parallel test execution at suite level, and the @Test methods will start running simultaneously. The thread count is set to be 2, which means that only two threads will be created for this execution, on which 2 test methods will start running parallely. If there are more than 2 methods, all the remaining ones need to wait until any of the threads get free.

Now that we know how to configure TestNG for running tests simultaneously, it’s time to do the hands-on! In the upcoming section of this Selenium TestNG tutorial, we will learn how to create a TestNG framework setup and write tests and execute them parallelly through the online Selenium Grid.

## Parallel Test Execution in multiple browsers using TestNG for Selenium automation testing

In this section, we will be writing test cases in the TestNG framework and learning how to execute them parallelly at tests, classes, and methods levels by configuring the TestNG XML file. Our test cases will be executed on 5 different browsers – Chrome, Firefox, Safari, IE, and Opera by leveraging the [online Selenium Grid](https://www.lambdatest.com/selenium-grid-online) provided by LambdaTest. Also, a custom configuration will be done for the browser as well as the OS version. Using LambdaTest, you can now test on [Safari browser online](https://www.lambdatest.com/test-on-safari-browsers) on both desktop and mobile browsers for [cross browser compatibility](https://www.lambdatest.com/learning-hub/cross-browser-compatibility).

If you are already working with TestNG, feel free to skip the installation and project set-up part, which we will be covering next.

### **Installation and Setup**

Before getting started with the project setup, make sure that your system is equipped with the following prerequisites:

1. **Java**

TestNG requires JDK 8 or higher. If you do not have Java installed in your system, you can [download Java](https://www.oracle.com/java/technologies/downloads/) from the Oracle official website. Don’t forget to set the environment variables as well.

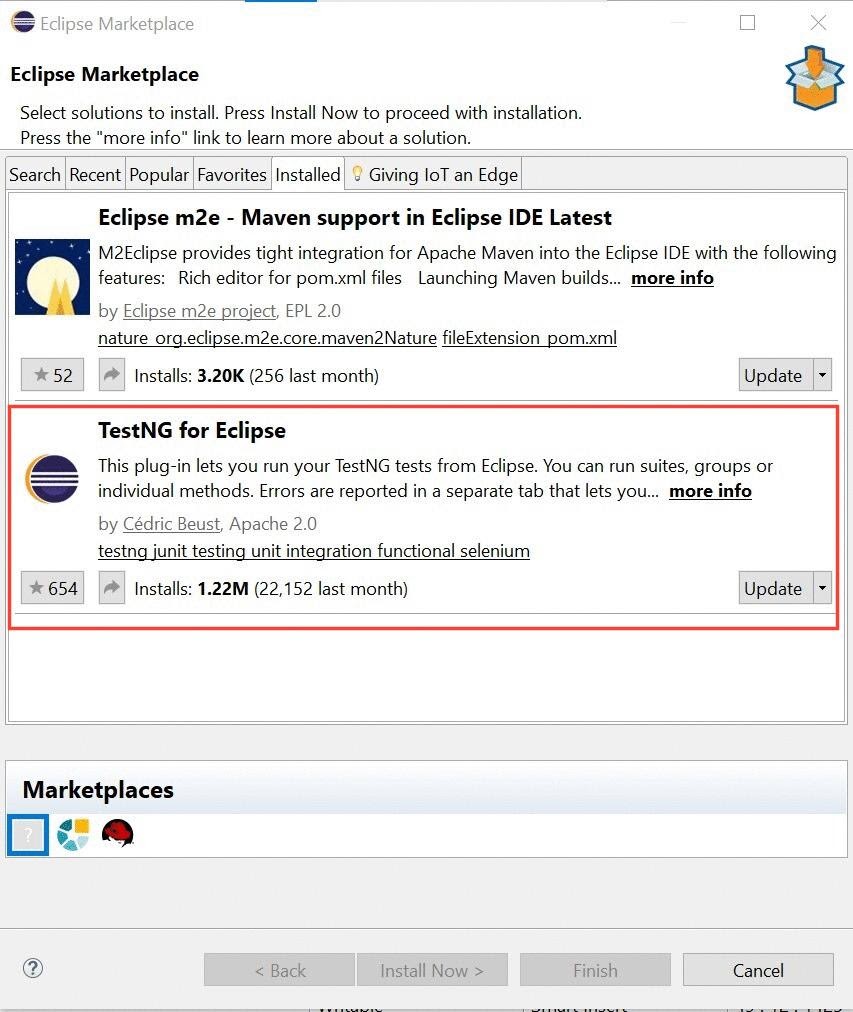
1. **Eclipse IDE**

In this tutorial, for the demo, we will be using Eclipse as IDE. If not installed already, you can [download Eclipse](https://www.eclipse.org/downloads/) from the official Eclipse website.

1. **Selenium Webdriver and Client for Java**

You can download by visiting the Selenium WebDriver [official website](https://www.selenium.dev/downloads/). [Selenium WebDriver](https://www.lambdatest.com/blog/selenium-webdriver-tutorial-with-examples/) is an open-source, free Selenium testing tool for browser automation.

Once this list is checked, the next step is to [install TestNG in Eclipse](https://www.lambdatest.com/blog/how-to-install-testng-in-eclipse-step-by-step-guide/). First, go to Eclipse Marketplace and install TestNG. Once installed, you can locate it under the Installed section.

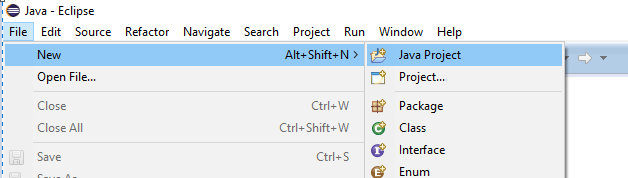


You have now completed the installation. Let us do the basic project setup now.

### **Creating a TestNG Project**

To use TestNG in a Java project, you must know [how to create a TestNG project](https://www.lambdatest.com/blog/create-testng-project-in-eclipse-run-selenium-test-script/). You can follow the below-mentioned steps to create a new Java project in Eclipse.

**Step 1**: Launch Eclipse and create a new Java project by navigating to **File > New > Java Project**.



**Step 2**: Name your project and click Next.

Graphical user interface, text, application, email

Description automatically generated

**Step 3**: Add libraries by following the below-mentioned steps:

1. Right-click on the project folder and select Properties.
2. Choose Java Build Path.
3. Go to Libraries, select Classpath, and click Add Library.

Graphical user interface, text, application

Description automatically generated

**Step 4**: Select TestNG and click Next.

Graphical user interface, text, application, Word

Description automatically generated

**Step 5**: Now, you will see that TestNG is added to the project library.

Graphical user interface, text, application, email

Description automatically generated

**Step 6**: Click Finish, and we have successfully created a TestNG project.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Step 7**: Add Selenium API JAR files to the TestNG project by following the below-mentioned steps:

1. Right-click on the project and select Properties.
2. Select Java Build Path.
3. Under the libraries section, click on Add External JARs, which gets enabled upon opting for Classpath.

Graphical user interface, text, application

Description automatically generated

1. Select the option and upload the JARs for Selenium Java language bindings by navigating to the path where you have downloaded and saved it.

Graphical user interface, text, application

Description automatically generated

1. Once done, select Apply and Close.

Graphical user interface, text, application, email

Description automatically generated

With this, our project set-up is done, and we are good to start coding. Notice that the libraries are also visible in the left panel.

Graphical user interface, text, application

Description automatically generated

### **How to run test methods in parallel with TestNG**

Now let us see how to run test methods parallely in the TestNG. In this example, we will create one more method and execute them on different threads. The steps are as below:

**Step 1**: Create a new TestNG class file. You can also opt for an auto-defined TestNG XML file.

Graphical user interface, text, application, email

Description automatically generated

**Step 2**: Inside this class, I have added a new test method forgotPassword() which will:

1. Launch the website URL (<https://opensource-demo.orangehrmlive.com/>).
2. Click on Forgot your password? link.
3. Enter Username.
4. Click the Reset Password button.

Graphical user interface, application

Description automatically generated



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | @Test    public void forgotPassword() {        // To navigate to URL      driver.get("https://opensource-demo.orangehrmlive.com/");      // To locate and click on Forgot password link      driver.findElement(By.linkText("Forgot your password?")).click();      // To locate input field and enter username      driver.findElement(By.id("securityAuthentication\_userName")).sendKeys("testuser");      // To locate and click on reset button      driver.findElement(By.id("btnSearchValues")).click();    } |

So the final code will look like this:

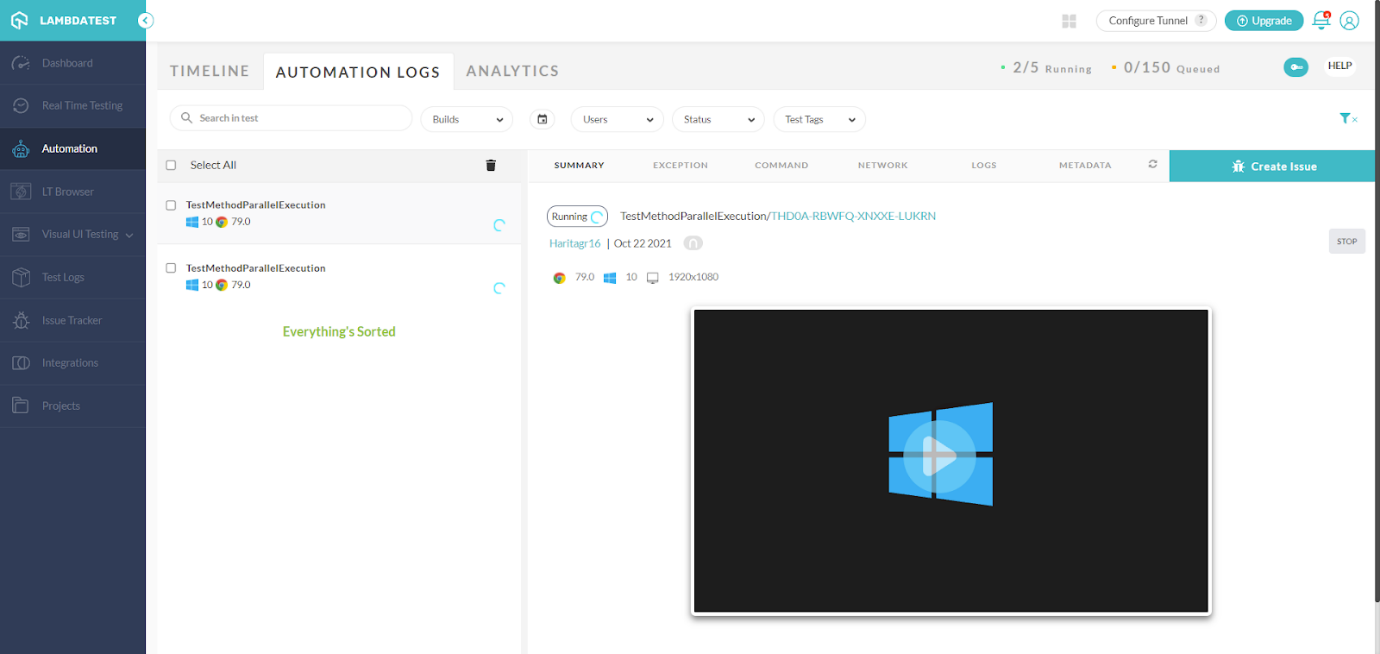
|  |  |
| --- | --- |
|  | package testng; |
|  | import java.net.MalformedURLException; |
|  | import java.net.URL; |
|  |  |
|  | import org.openqa.selenium.By; |
|  | import org.openqa.selenium.remote.DesiredCapabilities; |
|  | import org.openqa.selenium.remote.RemoteWebDriver; |
|  | import org.testng.annotations.AfterMethod; |
|  | import org.testng.annotations.BeforeMethod; |
|  | import org.testng.annotations.Parameters; |
|  | import org.testng.annotations.Test; |
|  |  |
|  | public class TestMethodParallelExecution { |
|  |  |
|  | public String username = "YOUR\_USERNAME"; |
|  | public String accesskey = "YOUR\_ACCESSKEY"; |
|  | public RemoteWebDriver driver = null; |
|  | public String gridURL = "@hub.lambdatest.com/wd/hub"; |
|  | boolean status = false; |
|  |  |
|  | @BeforeMethod |
|  | @Parameters(value={"browser","version","platform"}) |
|  | public void setUp(String browser, String version, String platform) throws Exception { |
|  | DesiredCapabilities capabilities = new DesiredCapabilities(); |
|  | capabilities.setCapability("browserName", browser); |
|  | capabilities.setCapability("version", version); |
|  | capabilities.setCapability("platform", platform); // If this cap isn't specified, it will just get the any available one |
|  | capabilities.setCapability("build", "TestMethodParallelExecution"); |
|  | capabilities.setCapability("name", "TestMethodParallelExecution"); |
|  | capabilities.setCapability("network", true); // To enable network logs |
|  | capabilities.setCapability("visual", true); // To enable step by step screenshot |
|  | capabilities.setCapability("video", true); // To enable video recording |
|  | capabilities.setCapability("console", true); // To capture console logs |
|  | try { |
|  | driver = new RemoteWebDriver(new URL("https://" + username + ":" + accesskey + gridURL), capabilities); |
|  | } catch (MalformedURLException e) { |
|  | System.out.println("Invalid grid URL"); |
|  | } catch (Exception e) { |
|  | System.out.println(e.getMessage()); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | @Test |
|  | public void login() { |
|  |  |
|  | // To navigate to URL |
|  | driver.get("https://opensource-demo.orangehrmlive.com/"); |
|  | // To locate input field for username and enter value |
|  | driver.findElement(By.name("txtUsername")).sendKeys("Admin"); |
|  | // To locate input field for password and enter value |
|  | driver.findElement(By.name("txtPassword")).sendKeys("admin123"); |
|  | // To locate and click on login button |
|  | driver.findElement(By.className("button")).click(); |
|  | } |
|  |  |
|  | @Test |
|  | public void forgotPassword() { |
|  |  |
|  | // To navigate to URL |
|  | driver.get("https://opensource-demo.orangehrmlive.com/"); |
|  | // To locate and click on Forgot password link |
|  | driver.findElement(By.linkText("Forgot your password?")).click(); |
|  | // To locate input field and enter username |
|  | driver.findElement(By.id("securityAuthentication\_userName")).sendKeys("testuser"); |
|  | // To locate and click on reset button |
|  | driver.findElement(By.id("btnSearchValues")).click(); |
|  | } |
|  |  |
|  | @AfterMethod |
|  | public void tearDown() throws Exception { |
|  | if (driver != null) { |
|  | driver.quit(); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | } |

[**view raw**](https://gist.github.com/muditlambda/701ea6d3396c9b7dd434bbacbe5796d9/raw/137e536c134ffcb5d01f27759c012e001cab9de4/TestMethodParallelExecution.java)[**TestMethodParallelExecution.java**](https://gist.github.com/muditlambda/701ea6d3396c9b7dd434bbacbe5796d9#file-testmethodparallelexecution-java)hosted with ❤ by [**GitHub**](https://github.com/)

**Step 3**: Configure the TestNG XML file. Set the value of the parallel attribute as ”methods”. My testng2.xml file now looks like this.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">  <suite name="Parallel Test Suite" parallel="methods" thread-count="2">        <test name="Chrome Test">          <parameter name="browser" value="chrome" />          <parameter name="version" value="79.0" />          <parameter name="platform" value="WIN10" />          <classes>              <class name="testng2.TestMethodParallelExecution" />          </classes>      </test>      <!-- Test -->  </suite> <!-- Suite --> |

**Step 4**: Run the tests. Our two test methods start running in parallel. You can observe the live running status in your LambdaTest dashboard.



### **How to run test classes in parallel with TestNG**

Lastly, let us see how to run test classes parallelly in the TestNG. First, let us create 2 classes that will be running parallel on separate threads.

1. Create 2 classes – TestClassOne and TestClassTwo. Each contains a test method.

**TestClassOne.java**

|  |  |
| --- | --- |
|  | package testng3; |
|  |  |
|  | import org.testng.annotations.Test; |
|  | import org.testng.annotations.BeforeMethod; |
|  | import org.testng.annotations.Parameters; |
|  |  |
|  | import java.net.MalformedURLException; |
|  | import java.net.URL; |
|  |  |
|  | import org.openqa.selenium.By; |
|  | import org.openqa.selenium.remote.DesiredCapabilities; |
|  | import org.openqa.selenium.remote.RemoteWebDriver; |
|  | import org.testng.annotations.AfterMethod; |
|  |  |
|  | public class TestClassOne { |
|  |  |
|  | public String username = "YOUR\_USERNAME"; |
|  | public String accesskey = "YOUR\_ACCESSKEY"; |
|  | public RemoteWebDriver driver = null; |
|  | public String gridURL = "@hub.lambdatest.com/wd/hub"; |
|  | boolean status = false; |
|  |  |
|  |  |
|  | @BeforeMethod |
|  | @Parameters(value={"browser","version","platform"}) |
|  | public void setUp(String browser, String version, String platform) throws Exception { |
|  | DesiredCapabilities capabilities = new DesiredCapabilities(); |
|  | capabilities.setCapability("browserName", browser); |
|  | capabilities.setCapability("version", version); |
|  | capabilities.setCapability("platform", platform); // If this cap isn't specified, it will just get the any available one |
|  | capabilities.setCapability("build", "TestMethodParallelExecution"); |
|  | capabilities.setCapability("name", "TestMethodParallelExecution"); |
|  | capabilities.setCapability("network", true); // To enable network logs |
|  | capabilities.setCapability("visual", true); // To enable step by step screenshot |
|  | capabilities.setCapability("video", true); // To enable video recording |
|  | capabilities.setCapability("console", true); // To capture console logs |
|  | try { |
|  | driver = new RemoteWebDriver(new URL("https://" + username + ":" + accesskey + gridURL), capabilities); |
|  | } catch (MalformedURLException e) { |
|  | System.out.println("Invalid grid URL"); |
|  | } catch (Exception e) { |
|  | System.out.println(e.getMessage()); |
|  | } |
|  | } |
|  |  |
|  | @Test |
|  | public void login() { |
|  |  |
|  | // To navigate to URL |
|  | driver.get("https://opensource-demo.orangehrmlive.com/"); |
|  | // To locate input field for username and enter value |
|  | driver.findElement(By.name("txtUsername")).sendKeys("Admin"); |
|  | // To locate input field for password and enter value |
|  | driver.findElement(By.name("txtPassword")).sendKeys("admin123"); |
|  | // To locate and click on login button |
|  | driver.findElement(By.className("button")).click(); |
|  | } |
|  |  |
|  | @AfterMethod |
|  | public void tearDown() throws Exception { |
|  | if (driver != null) { |
|  | System.out.println("Tearing down.."); |
|  | driver.quit(); |
|  | } |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/muditlambda/0dfa5129f85e6746a49ec623aec8546d/raw/cdb8b9ca9108ad3626a164d4efb239f0dc8ea46f/TestClassOne.java)[**TestClassOne.java**](https://gist.github.com/muditlambda/0dfa5129f85e6746a49ec623aec8546d#file-testclassone-java)hosted with ❤ by [**GitHub**](https://github.com/)

**TestClassTwo.java**

|  |  |
| --- | --- |
|  | package testng3; |
|  |  |
|  | import org.testng.annotations.Test; |
|  | import org.testng.annotations.BeforeMethod; |
|  | import org.testng.annotations.Parameters; |
|  |  |
|  | import java.net.MalformedURLException; |
|  | import java.net.URL; |
|  |  |
|  | import org.openqa.selenium.By; |
|  | import org.openqa.selenium.remote.DesiredCapabilities; |
|  | import org.openqa.selenium.remote.RemoteWebDriver; |
|  | import org.testng.annotations.AfterMethod; |
|  |  |
|  | public class TestClassTwo { |
|  |  |
|  | public String username = "haritagr16"; |
|  | public String accesskey = "KhRUhWBtSMg06DfGkKDY1khYNJm95695WV42HfaYPot7nqQfw6"; |
|  | public RemoteWebDriver driver = null; |
|  | public String gridURL = "@hub.lambdatest.com/wd/hub"; |
|  | boolean status = false; |
|  |  |
|  | @BeforeMethod |
|  | @Parameters(value={"browser","version","platform"}) |
|  | public void setUp(String browser, String version, String platform) throws Exception { |
|  | DesiredCapabilities capabilities = new DesiredCapabilities(); |
|  | capabilities.setCapability("browserName", browser); |
|  | capabilities.setCapability("version", version); |
|  | capabilities.setCapability("platform", platform); // If this cap isn't specified, it will just get the any available one |
|  | capabilities.setCapability("build", "ParallelTestNG"); |
|  | capabilities.setCapability("name", "ParallelTestNG"); |
|  | capabilities.setCapability("network", true); // To enable network logs |
|  | capabilities.setCapability("visual", true); // To enable step by step screenshot |
|  | capabilities.setCapability("video", true); // To enable video recording |
|  | capabilities.setCapability("console", true); // To capture console logs |
|  | try { |
|  | driver = new RemoteWebDriver(new URL("https://" + username + ":" + accesskey + gridURL), capabilities); |
|  | } catch (MalformedURLException e) { |
|  | System.out.println("Invalid grid URL"); |
|  | } catch (Exception e) { |
|  | System.out.println(e.getMessage()); |
|  | } |
|  | } |
|  |  |
|  | @Test |
|  | public void forgotPassword() { |
|  |  |
|  | // To navigate to URL |
|  | driver.get("https://opensource-demo.orangehrmlive.com/"); |
|  | // To locate and click on Forgot password link |
|  | driver.findElement(By.linkText("Forgot your password?")).click(); |
|  | // To locate input field and enter username |
|  | driver.findElement(By.id("securityAuthentication\_userName")).sendKeys("testuser"); |
|  | // To locate and click on reset button |
|  | driver.findElement(By.id("btnSearchValues")).click(); |
|  | } |
|  |  |
|  | @AfterMethod |
|  | public void tearDown() throws Exception { |
|  | if (driver != null) { |
|  | System.out.println("Tearing down.."); |
|  | driver.quit(); |
|  | } |
|  | } |
|  | } |

2. Create a new TestNG XML file and set the parallel attribute as “classes.”

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">  <suite name="Parallel Test Suite" parallel="classes" thread-count="2">        <test name="Chrome Test">          <parameter name="browser" value="chrome" />          <parameter name="version" value="79.0" />          <parameter name="platform" value="WIN10" />          <classes>              <class name="testng3.TestClassOne" />              <class name="testng3.TestClassTwo" />          </classes>      </test>      <!-- Test -->  </suite> <!-- Suite --> |

3. Run your test suite. The test gets executed successfully.

