TestNG Parallel Test Execution XML

Running tests in **parallel using XML** typically refers to executing tests concurrently in test frameworks like **TestNG** (commonly used with **Java + Selenium**). TestNG allows you to define and control parallel execution in its **`testng.xml`** file.

✅ Step-by-Step Guide to Run Tests in Parallel Using `testng.xml`

1. Structure of `testng.xml` for Parallel Execution

```xml
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="ParallelSuite" parallel="tests" thread-count="2">
    <test name="ChromeTest">
        <parameter name="browser" value="chrome"/>
        <classes>
            <class name="com.tests.TestClass1"/>
        </classes>
    </test>

    <test name="FirefoxTest">
        <parameter name="browser" value="firefox"/>
        <classes>
            <class name="com.tests.TestClass2"/>
        </classes>
    </test>
</suite>
```

🔍 Key Attributes

Attribute	Description
`parallel="tests"`	Executes **`<test>`** tags in parallel
`thread-count="2"`	Maximum number of threads to run simultaneously
`parameter`	Used to pass browser type or other parameters to test classes

---

🪛 Types of Parallel Execution in TestNG

You can change the **`parallel`** attribute to one of the following:

Mode	Description
`tests`	Runs **`<test>`** blocks in parallel
`classes`	Runs test classes in parallel
`methods`	Runs individual test methods in parallel
`instances`	Runs test instances in parallel

✅ Example for Parallel Methods

```xml
<suite name="ParallelSuite" parallel="methods" thread-count="3">
    <test name="TestMethods">
        <classes>
            <class name="com.tests.ParallelTestClass"/>
        </classes>
    </test>
</suite>
```

And the Java class:

```java
public class ParallelTestClass {

    @Test
    public void test1() {
        System.out.println("Test1 - Thread ID: " + Thread.currentThread().getId());
    }

    @Test
    public void test2() {
        System.out.println("Test2 - Thread ID: " + Thread.currentThread().getId());
    }

    @Test
    public void test3() {
        System.out.println("Test3 - Thread ID: " + Thread.currentThread().getId());
    }
}
```

📌 Important Notes

1. **Thread Safety**: Ensure WebDriver or any shared resource is thread-safe when running parallel tests.
2. **Parallel Browsers**: Combine with Selenium Grid or use multiple driver instances (like ChromeDriver and FirefoxDriver) for parallel cross-browser testing.
3. **TestNG `@Parameters` or `@DataProvider(parallel = true)`**: Can also be used for dynamic and scalable parallel execution.

Would you like me to show how to integrate this with Selenium WebDriver or generate sample classes with full code?

Great! Let's go deeper into **`thread`** handling in the context of **parallel test execution using TestNG and XML configuration**.

🧵 What Is a Thread in TestNG?

In parallel testing, a **thread** represents a separate path of execution. **Multiple threads** allow you to **run multiple tests at the same time**, which speeds up the execution process—especially helpful for large test suites or cross-browser testing.

✅ Thread Attributes in TestNG XML

🔹 `thread-count` (in `testng.xml`)

This attribute controls **how many threads** will run **in parallel**.

```xml
xml

<suite name="ParallelSuite" parallel="tests" thread-count="3">
```

- Here, **`thread-count="3"`** means a maximum of 3 test threads can run **concurrently**.
- It is applied at the **`<suite>`** level.

🔹 `parallel` Options

```xml
xml

parallel="tests"       <!-- Runs test blocks in parallel -->
parallel="classes"     <!-- Runs classes in parallel -->
parallel="methods"     <!-- Runs test methods in parallel -->
```

Each of these dictates **what level** of test structure will be executed on **separate threads**.

🔪 Thread Example: Parallel by Class

testng.xml

```xml
xml

<suite name="ParallelClasses" parallel="classes" thread-count="2">
    <test name="ClassLevelParallel">
        <classes>
            <class name="com.tests.TestClassA"/>
            <class name="com.tests.TestClassB"/>
        </classes>
    </test>
</suite>
```

Java Classes

```java
public class TestClassA {
    @Test
    public void testA1() {
        System.out.println("TestA1 - Thread: " + Thread.currentThread().getId());
    }
}

public class TestClassB {
    @Test
    public void testB1() {
        System.out.println("TestB1 - Thread: " + Thread.currentThread().getId());
    }
}
```

Sample Output:

```yaml
TestA1 - Thread: 12
TestB1 - Thread: 13
```

This confirms both test classes are running on **separate threads**.

🔒 Thread Safety in Selenium Tests

When using Selenium WebDriver in parallel threads:

- **Don't share WebDriver instances** between threads.
- Use **ThreadLocal** WebDriver pattern or create **a new WebDriver instance per thread**.

Example: ThreadLocal WebDriver

```java
public class DriverFactory {
    private static ThreadLocal<WebDriver> driver = new ThreadLocal<>();

    public static WebDriver getDriver() {
        return driver.get();
    }

    public static void setDriver(WebDriver driverInstance) {
        driver.set(driverInstance);
    }

    public static void quitDriver() {
        getDriver().quit();
        driver.remove();
    }
}
```

This ensures **each thread** gets its **own isolated WebDriver**.

✅ Summary: Key Points on Threads

Element	Purpose
`thread-count`	Maximum number of threads to run in parallel
`parallel`	Specifies test level to run in parallel
Thread Safety	Each thread must manage its own WebDriver
Speed Benefit	Reduced execution time for large test suites

Would you like me to show a full working Selenium + TestNG parallel example with **`ThreadLocal`** and XML config?