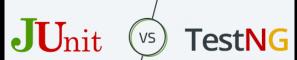
JUnit (vs) TestNG

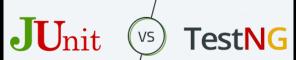


ription	TestNG (6.10)	Junit (Junit 5)	
Execute Before / After each suite	@BeforeSuite @AfterSuite	Not Available	
Execute Before / After each class	@BeforeClass	@BeforeAll	
	@AfterClass	@AfterAll	
	- KA1	Executed Before/ After all @Test,	
		@RepeatedTest, @ParameterizedTest, and	
	2 D A 2	@TestFactory methods in the current class;	
Evacuta Rafara /	@DefereMethod	analogous to JUnit 4's @BeforeClass.	
		Not Available	
Execute Before /		@BeforeEach	
After each Test	@AfterTest	@AfterEach	
	200	Executed before each @Test,	
	- KA	@RepeatedTest, @ParameterizedTest, or	
	1 100000	@TestFactory method in the current class;	
	2 DA 2 DA 2 DA 2	analogous to JUnit 4's @Before	
· ·		Not Available	
After each group	@AfterGroups		
	@Test	@Test	
Description	@Test(description='Login Test')	@Test	
		<pre>@DisplayName("Custom test name containing spaces")</pre>	
		***Inherit the 'DisplayNameGenerator.	
		ReplaceUnderscores' class to update the	
	- DA DA2	report format	
Sequential @Test(priority=1)		@Test	
Execution		@ Order(1)	
Execute one test	@Test(dependsOnMethods = 'methodName')	Not Available	
•	@Test(groups='Regression')	@Tag("Regression")	
		@Test	
ignore rest	C rest (chashed lates)	@Disabled('Execute after fix')	
	FRAF	@EnabledOnOs({ LINUX, MAC })	
		@DisabledOnOs(WINDOWS)	
		@EnabledIfSystemProperty(named =	
		"os.arch", matches="true")	
		@DisabledIfSystemProperty(named = "ci-	
	1	server", matches = "true")	
	Execute Before / After each suite Execute Before / After each class Execute Before / After each Method Execute Before / After each Test Execute Before / After each group Test Method Description Sequential Execution	Execute Before / After each suite Execute Before / After each class Execute Before / After each class Execute Before / After each Method Execute Before / After each Method Execute Before / After each Test Execute Before / After each Test Execute Before / After each Test Execute Before / After each group Execute Before / After each Test Execute Before / After Pack Exe	





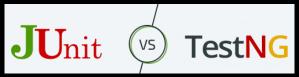
	ework overview		
escription	TestNG (6.10)	Junit (Junit 5)	
	² RA ² RA ²	@EnabledIfEnvironmentVariable(named = "ENV", matches = "staging-server") @DisabledIfEnvironmentVariable(named = "ENV", matches = ".*development.*")	
	² RA ²	@EnabledIf("2 * 3 == 6") @DisabledIf("2 * 3 == 6")	
Time Out	@Test (timeout = 100)	Not Available	
Exception	<pre>@Test(expectedExceptions = ArithmeticExpeption.class)</pre>	Not Available	
Execute the same	@Test(InvocationCount=10 ,threadPollSize=5,	@RepeatedTest(10)	
test at multiple times	invocationTimeOut=10, successPercentage=50) (timeout in ms) (percentage of success expected from this method)	<pre>@RepeatedTest(value = 1, name = "{displayName} {currentRepetition}/{totalRepetitions}") @DisplayName("Repeat!")</pre>	
Always runs even if the parameters on which the method depends fails.	@Test(alwaysRun = true)	Not Available	
Class level Annotation	<pre>@Test(groups = { "checkin-test" })</pre>	@Disabled("Disabled until this bug#11 fixed") @Tag("Disabled")	
	public class All {	class DisabledClassDemo {	
	<pre>@Test(groups = { "func-test") public void method1() { } public void method2() { } } **Method – part of both group, method2 is part of 'checkin-test' ** Method 1 and 2 are Test Methods</pre>	<pre>@Test Public void testWillBeSkipped() { } }</pre>	
TestNG -	@DataProvider(name = "test1")	@ParameterizedTest	
[@DataProvider] Supply the test data to test case using @dataprovider	<pre>public static Object[][] validUserName() { Read the value from excel / DB /othes and</pre>	@ValueSource(strings = { "racecar", "radar", "able was I ere I saw elba" })	
using		@ValueSource(ints = { 1, 2, 3 })	
@dataprovider annotation,	storied in the double dimension object }	@ValueSource(strings = "SECONDS")	





Des	cription	TestNG (6.10)	Junit (Junit 5)		
	To execute same test in multiple times with	@Test(dataProvider = "test1", dataProviderClass = 'validUserName .class ')	@EnumSource(value = TimeUnit.class, names = { "DAYS", "HOURS" })		
	different data.	<pre>public void verifyValidUserName(String UserName) { }</pre>	@MethodSource("stringProvider")		
	Test NG - [@Parameters] Supply the test data from TestNG.xml to test case using @parameters annotation	<pre>@Parameters("Name") @Test public void verifyValidUserName(String UserName) { } TestNG Suite File (xml) <suite name="Suite1"></suite></pre>	@CsvSource({ "apple, banana", "Orange, Mango" }) @CsvFileSource(resources = "/two-column.csv", numLinesToSkip = 1)		
Assertion	Assertion to validate the actual with expected condition.	Assert.assertEquals(actual,expected); Assert.assertNotEquals(actual,expected,Message); Assert.assertTrue(condition); Assert.assertFalse(condition); Assert.assertNull(object); Assert.assertNotNull(object);	<pre>import static org.junit.jupiter.api.Assertions. assertEquals; assertEquals(actual, expected); assertEquals(actual, expected); assertEquals(actual, expected, Message); assertTrue(condition, message); assertFalse(condition, message); assertAll("last name", () -> assertTrue(lastName.startsWith("D")), () -> assertTrue(lastName.endsWith("e"))); Assert.assertNull(object); Assert.assertNotNull(object); assertTimeout(ofMillis(10), () -> { // Simulate task that takes more than 10 ms. Thread.sleep(100); });</pre>		
ш		<suite name="My test suite"></suite>	@RunWith(JUnitPlatform.class)		





Description	TestNG (6.10)	Junit (Junit 5)		
Execute the automation test suite using test framework.	<test name="testing"> <classes> <class name="com.fsecure.demo.testng.Test1"></class> <class name="com.fsecure.demo.testng.Test2"></class> </classes></test>	<pre>@SelectPackages({"com.howtodoinjava.junit 5.e xamples.packageA","com.howtodoinjava.juni t5.examples.packageB"}) public class JUnit5TestSuiteExample { } Single / Multiple Class @SelectClasses({ ClassATest.class,</pre>		
	<class name="com.easy.entry.AddTestCase"> <methods> <include name="addLocTestCase"></include> <exclude name="addEmplTestCase"></exclude> </methods> </class> <test name="Simple example"> <groups> <run> <include name="checkintest"></include> <exclude name="broken"></exclude> </run> </groups></test>	Include / Exclude Package @SelectPackages("com.howtodoinjava.junit5 .examples") @IncludePackages("com.howtodoinjava.junit 5.examples.packageC") @ExcludePackages("com.howtodoinjava.junit 5.examples.packageC") Include / Exclude class @IncludeClassNamePatterns({"^.*ATests?\$"}) @ExcludeClassNamePatterns({"^.*ATests?\$"}		
		Include/ Exclude Tags @IncludeTags("production") @ExcludeTags("development") JUNIT 4 @RunWith(Suite.class) @Suite.SuiteClasses({ JunitTest1.class, JunitTest2.class		
Execute Tests in parallel	Suite Level Parallel Execution cmd >> java org.testng.TestNG - suitethreadpoolsize 3 testng1.xml testng2.xml testng3.xml	public class JunitTest5 { } Resources -junit-platform.properties file junit.jupiter.execution.parallel.enabled=true junit.jupiter.execution.parallel.config.strategy =dynamic		

JUnit (vs) TestNG



Description	TestNG (6.10)	Junit (Junit 5)
	Method Level Parallel Execution <suite name="My suite" parallel="methods" thread-count="5"></suite>	@Execution(ExecutionMode.CONCURRE NT)
	Tests Level Parallel Execution <suite name="My suite" parallel="tests" thread-count="5"> Class Level Parallel Execution <suite name="My suite" parallel="classes" thread-count="5"> Instances Level Parallel Execution <suite name="My suite" parallel="instances" thread-count="5"> Instances Level Parallel Execution <suite name="My suite" parallel="instances" thread-count="5"> Instances Level Parallel Execution <suite name="My suite" parallel="instances" thread-count="5"> Instances Level Parallel Execution <suite name="My suite" parallel="instances" thread-count="5"> Instances Level Parallel Execution <suite name="My suite" parallel="instances" thread-count="5"> Instances Level Parallel Execution <suite name="My suite" parallel="instances" thread-count="5"> Instances Level Parallel Execution <suite name="My suite" parallel="instances" thread-count="5"> Instances Level Parallel Execution <suite name="My suite" parallel="instances" thread-count="5"> Instances Level Parallel Execution <suite name="My suite" parallel="instances" thread-count="5"> Instances Level Parallel Execution <suite name="My suite" parallel="instances" thread-count="5"> Instances Level Parallel Execution <suite name="My suite" parallel="instances" thread-count="5"> Instances Level Parallel Execution Instances Level P</suite></suite></suite></suite></suite></suite></suite></suite></suite></suite></suite></suite></suite>	<pre>@Execution(ExecutionMode.CONCURRENT) public class LoginTest extends BaseTest { @Test public void invalidLoginTest_(){ } @Test public void invalidLoginTest_(){ } } SAME_THREAD Force execution in the same thread used by the parent. For example, when used on a test method, the test method will be executed in the same thread as any @BeforeAll or @AfterAll methods of the containing test class. CONCURRENT Execute concurrently unless a resource lock forces execution in the same thread.</pre>
Listeners	IAnnotationTransformer IAnnotationTransformer2 IHookable IInvokedMethodListener IMethodInterceptor IReporter ISuiteListener ITestListener <suite></suite>	Listener TestExecutionListener SummaryGeneratingListener LegacyXmlReportGeneratingListener Extension Model



JUnit (vs) TestNG

_				•
Test	LKOK	\mathbf{a}		verview
1671			/ (C) K	VEIVIEW

Description	TestNG (6.10)	Junit (Junit 5)	
		@ExtendWith(TestLifeCycleExtensions.class)	
	listener class-	Public class TestInstancePostProcessor,	
	name="com.example.MyListener" />	BeforeAllCallback, BeforeEachCallback,	
	listener class-	BeforeTestExecutionCallback,	
	name="com.example.MyMethodInterceptor"	AfterTestExecutionCallback,	
	/>	AfterEachCallback, AfterAllCallback{	
		//Override & modify as per our need }	
	² RA ³	<pre>@ExtendWith(TestLifeCycleExtensions.class) public class test1 {</pre>	
	2 RA:	RA ² F	

