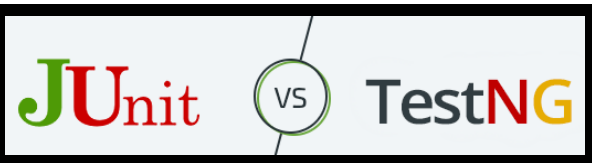
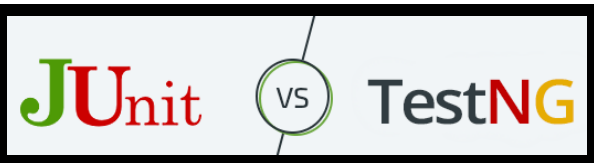
Test Framework Overview



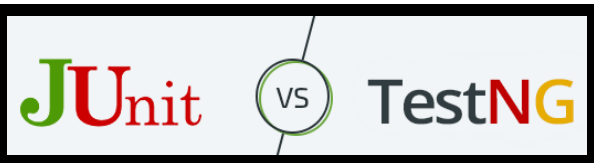
Description		TestNG (6.10)	Junit (Junit 5)
Prerequisite	Execute Before / After each suite	@BeforeSuite @AfterSuite	**Not Available**
	Execute Before / After each class	@BeforeClass @AfterClass	@BeforeAll @AfterAll
			Executed Before/ After all @Test, @RepeatedTest, @ParameterizedTest, and @TestFactory methods in the current class; analogous to JUnit 4's @BeforeClass.
	Execute Before / After each Method	@BeforeMethod @AfterMethod	**Not Available**
	Execute Before / After each Test	@BeforeTest @AfterTest	@BeforeEach @AfterEach
			Executed before each @Test, @RepeatedTest, @ParameterizedTest, or @TestFactory method in the current class; analogous to JUnit 4's @Before
	Execute Before / After each group	@BeforeGroups @AfterGroups	**Not Available**
Test Annotation	Test Method	@Test	@Test
	Description	@Test(description='Login Test')	@Test *@DisplayName*("Custom test name containing spaces")
			***Inherit the '*DisplayNameGenerator.ReplaceUnderscores*' class to update the report format
	Sequential Execution	@Test(priority=1)	@Test @ Order(1)
	Execute one test after based previous result	@Test(dependsOnMethods = 'methodName')	**Not Available**
	Grouping the Test	@Test(groups='Regression')	@Tag("Regression")
	Ignore Test	@Test (enabled =false)	@Test
			@Disabled('Execute after fix')
			@EnabledOnOs({ LINUX, MAC }) *@DisabledOnOs*(WINDOWS)
			@EnabledIfSystemProperty(named = "os.arch", matches="true") *@DisabledIfSystemProperty*(named = "ci-server", matches = "true")
			@EnabledIfEnvironmentVariable(named = "ENV", matches = "staging-server") *@DisabledIfEnvironmentVariable*(named = "ENV", matches = ".*development.*")

Test Framework Overview



Description		TestNG (6.10)	Junit (Junit 5)
			@EnabledIf("2 * 3 == 6") *@DisabledIf*("2 * 3 == 6")
	Time Out	@Test (timeout = 100)	**Not Available**
	Exception	@Test(expectedExceptions = ArithmeticExpeption.class)	**Not Available**
	Execute the same test at multiple times	@Test(InvocationCount=10 ,threadPollSize=5, invocationTimeOut=10, successPercentage=50) (timeout in ms) (percentage of success expected from this method)	**@RepeatedTest(10)** **@RepeatedTest**(value = 1, name = "{displayName} {currentRepetition}/{totalRepetitions}") **@DisplayName("Repeat!")**
	Always runs even if the parameters on which the method depends fails.	@Test(alwaysRun = true)	**Not Available**
	Class level Annotation	**@Test(groups = { "checkin-test" })** public class All { *@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*	**@Disabled("Disabled until this bug#11 fixed")** **@Tag("Disabled")** class DisabledClassDemo { @Test Public void testWillBeSkipped() { } }
Parameterization	**TestNG - [@DataProvider]** Supply the test data to test case using @dataprovider annotation, To execute same test in multiple times with different data.	**@DataProvider(name = "test1")** *public static Object[][] validUserName() {* *Read the value from excel / DB /othes and storied in the double dimension object* } **@Test(dataProvider = "test1", dataProviderClass = '*validUserName* .class ')** public void verifyValidUserName(String UserName) { }	**@ParameterizedTest** **@*ValueSource***(strings = { "racecar", "radar", "able was I ere I saw elba" }) *@ValueSource*(ints = { 1, 2, 3 }) *@ValueSource*(strings = "SECONDS") **@*EnumSource***(value = TimeUnit.class, names = { "DAYS", "HOURS" }) *@MethodSource*("stringProvider") **@*CsvSource***({ "apple, banana" , "Orange, Mango" })
	Test NG - [@Parameters]	**@Parameters("Name")** @Test	

Test Framework Overview



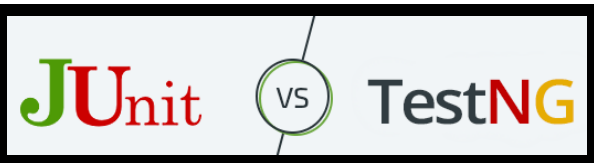
Description		TestNG (6.10)	Junit (Junit 5)
	Supply the test data from TestNG.xml to test case using @parameters annotation	public void verifyValidUserName(String UserName) { } **TestNG Suite File (xml)** <suite name = "Suite1"> <test name = "test1"> **<parameter name = "Name" value="user01"/>** <classes> <class name = "ParameterizedTest1" /> </classes> </test> </suite>	***@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);	**import static org.junit.jupiter.api.Assertions.assertEquals;** assertEquals(actual, expected); assertNotEquals(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); });
Execution	Execute the automation test suite using test framework.	<suite name="My test suite"> <test name="testing"> <classes> <class name="com.fsecure.demo.testng.Test1"/> <class name="com.fsecure.demo.testng.Test2"/> </classes> </test> </suite> <class name="com.easy.entry.AddTestCase"> <methods> <include name="addLocTestCase" /> <exclude name="addEmplTestCase" /> </methods> </class>	**@RunWith(JUnitPlatform.class)** @SelectPackages({"com.howtodoinjava.junit5.examples.packageA","com.howtodoinjava.junit5.examples.packageB"}) public class JUnit5TestSuiteExample { } **Single / Multiple Class** @SelectClasses({ ClassATest.class, ClassBTest.class, ClassCTest.class }) **Include / Exclude Package** @SelectPackages("com.howtodoinjava.junit5.examples") @IncludePackages("com.howtodoinjava.junit5.examples.packageC")

Test Framework Overview

JUnit VS TestNG

Description		TestNG (6.10)	Junit (Junit 5)
		<test name="Simple example"> <groups> <run> <include name="checkintest"/> <exclude name="broken"/> </run> </groups>	@ExcludePackages("com.howtodoinjava.junit5.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 }) public class JunitTest5 { }
Parallel Execution	Execute Tests in parallel	***Suite Level Parallel Execution*** cmd >> java org.testng.TestNG -suitethreadpoolsize 3 testng1.xml testng2.xml testng3.xml	**Resources -junit-platform.properties file** junit.jupiter.execution.parallel.enabled=true junit.jupiter.execution.parallel.config.strategy=dynamic
		Method Level Parallel Execution <suite name="My suite" parallel="methods" thread-count="5">	**@Execution(ExecutionMode.CONCURRENT)** *@Execution(ExecutionMode.CONCURRENT)* *public class LoginTest extends BaseTest {* *@Test* *public void invalidLoginTest_(){ }* *@Test* *public void invalidLoginTest_(){ }* *}*
		Tests Level Parallel Execution <suite name="My suite" parallel="tests" thread-count="5">	***SAME_THREAD***
		Class Level Parallel Execution <suite name="My suite" parallel="classes" thread-count="5">	
		Instances Level Parallel Execution	

Test Framework Overview



Description		TestNG (6.10)	Junit (Junit 5)
		<suite name="My suite" parallel="instances" thread-count="5">	*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.*
Listeners		IAnnotationTransformer	**Listener**
		IAnnotationTransformer2	**TestExecutionListener**
		IHookable	SummaryGeneratingListener
		IInvokedMethodListener	LegacyXmlReportGeneratingListener
		IMethodInterceptor	
		IReporter	
		ISuiteListener	
		ITestListener	
		<suite> <listeners> <listener class-name="com.example.MyListener" /> <listener class-name="com.example.MyMethodInterceptor" /> </listeners>	**Extension Model** **@ExtendWith(TestLifeCycleExtensions.class)** Public class TestInstancePostProcessor, BeforeAllCallback, BeforeEachCallback, BeforeTestExecutionCallback, AfterTestExecutionCallback, AfterEachCallback, AfterAllCallback{ //Override & modify as per our need } @ExtendWith(TestLifeCycleExtensions.class) public class test1 { }