| **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.\*") |
| @*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" })** | **@Disabled("Disabled until this bug#11 fixed")**  **@Tag(“Disabled”)** |
| 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* | 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")** | **@ParameterizedTest** |
| **@*ValueSource***(strings = { "racecar", "radar", "able was I ere I saw elba" }) |
| *public static Object[][] validUserName() {*  *Read the value from excel / DB /othes and storied in the double dimension object*  } |
| @*ValueSource*(ints = { 1, 2, 3 }) |
| @*ValueSource*(strings = "SECONDS") |
| **@Test(dataProvider = "test1", dataProviderClass = ‘*validUserName* .class ‘)** |
| **@*EnumSource***(value = TimeUnit.class, names = { "DAYS", "HOURS" }) |
| public void verifyValidUserName(String UserName) {  } |
| @*MethodSource*("stringProvider") |
| **@*CsvSource***({ "apple, banana" , "Orange, Mango" }) |
| **Test NG - [@Parameters]**  Supply the test data from TestNG.xml to test case using @parameters annotation | **@Parameters("Name")** |
| @Test  public void verifyValidUserName(String UserName) {  } |
| **@*CsvFileSource*(**resources = "/two-column.csv", numLinesToSkip = 1) |
| **TestNG Suite File (xml)**  <suite name = "Suite1">  <test name = "test1"> |
| **<parameter name = “Name" value="user01"/>** |
| <classes>  <class name = "ParameterizedTest1" />  </classes>  </test>  </suite> |
| **Assertion** | Assertion to validate the actual with expected condition. | Assert.assertEquals(actual,expected); | **import static org.junit.jupiter.api.Assertions.** **assertEquals;** |
| assertEquals(actual, expected); |
| Assert.assertNotEquals(actual,expected,Message); | assertNotEquals(actual, expected); |
| assertEquals(actual, expected, Message); |
| Assert.assertTrue(condition); | assertTrue(condition, message); |
| Assert.assertFalse(condition); | assertFalse(condition, message); |
| Assert.assertNull(object); | assertAll("last name",  () -> assertTrue(lastName.startsWith("D")),  () -> assertTrue(lastName.endsWith("e")) ); |
| Assert.assertNotNull(object); |
| 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> | **@RunWith(JUnitPlatform.class)** |
| @SelectPackages({"com.howtodoinjava.junit5.e xamples.packageA","com.howtodoinjava.junit5.examples.packageB"})  public class JUnit5TestSuiteExample  {  } |
| **Single / Multiple Class**  @SelectClasses( { ClassATest.class, ClassBTest.class, ClassCTest.class } ) |
| <**class** name="com.easy.entry.AddTestCase"> <**methods**>  <**include** name="addLocTestCase" />  <**exclude** name="addEmplTestCase" />  </**methods**>  </**class**> |
| **Include / Exclude Package**  @SelectPackages("com.howtodoinjava.junit5.examples")  @IncludePackages("com.howtodoinjava.junit5.examples.packageC")  @ExcludePackages("com.howtodoinjava.junit5.examples.packageC") |
| <test name="Simple example">  <groups>  <run>  <include name="checkintest"/>  <exclude name="broken"/>  </run>  </groups> |
| **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*** | **Resources -junit-platform.properties file** |
| cmd >> java org.testng.TestNG -suitethreadpoolsize 3 testng1.xml testng2.xml testng3.xml | 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\_(){ }*  *}*  ***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.* |
| ***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"> |
| **Listeners** |  | IAnnotationTransformer | **Listener** |
| IAnnotationTransformer2 | **TestExecutionListener**  SummaryGeneratingListener  LegacyXmlReportGeneratingListener |
| IHookable |
| IInvokedMethodListener |
| 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 {  } |