JUnit 4 and TestNG are both very popular unit test framework in Java. Both frameworks look very similar in functionality. Which one is better? Which unit test framework should I use in Java project? Here I did a feature comparison between JUnit 4 and TestNG.

**1. Annotation Support**

|  |  |  |
| --- | --- | --- |
| Feature | JUnit 4 | TestNG |
| test annotation | @Test | @Test |
| run before all tests in this suite have run | — | @BeforeSuite |
| run after all tests in this suite have run | — | @AfterSuite |
| run before the test | — | @BeforeTest |
| run after the test | — | @AfterTest |
| run before the first test method that belongs to any of these groups is invoked | — | @BeforeGroups |
| run after the last test method that belongs to any of these groups is invoked | — | @AfterGroups |
| run before the first test method in the current class is invoked | @BeforeClass | @BeforeClass |
| run after all the test methods in the current class have been run | @AfterClass | @AfterClass |
| run before each test method | @Before | @BeforeMethod |
| run after each test method | @After | @AfterMethod |
| ignore test | @ignore | @Test(enbale=false) |
| expected exception | @Test(expected = ArithmeticException.class) | @Test(expectedExceptions = ArithmeticException.class) |
| timeout | @Test(timeout = 1000) | @Test(timeout = 1000) |

The main annotation differences between JUnit4 and TestNG are

1. In JUnit 4, we have to declare “@BeforeClass” and “@AfterClass” method as static method. TestNG is more flexible in method declaration, it does not have this constraints.

2. 3 additional setUp/tearDown level: suite and group (@Before/AfterSuite, @Before/AfterTest, @Before/AfterGroup).

## 2. Exception Test

The “exception testing” means what exception throws from the unit test, this feature is implemented in both JUnit 4 and TestNG.

## 3. Ignore Test

The “Ignored” means whether it should ignore the unit test, this feature is implemented in both JUnit 4 and TestNG .

## 4. Time Test

The “Time Test” means if an unit test takes longer than the specified number of milliseconds to run, the test will terminated and mark as fails, this feature is implemented in both JUnit 4 and TestNG .

## 5. Suite Test

The “Suite Test” means bundle a few unit test and run it together. This feature is implemented in both JUnit 4 and TestNG. However both are using very different method to implement it.

JUnit 4

The “@RunWith” and “@Suite” are use to run the suite test. The below class means both unit test “JunitTest1” and “JunitTest2” run together after JunitTest5 executed. All the declaration is define inside the class.

TestNG

XML file is use to run the suite test. The below XML file means both unit test “TestNGTest1” and “TestNGTest2” will run it together.

TestNG can do more than bundle class testing, it can bundle method testing as well. With TestNG unique “Grouping” concept, every method is tie to a group, it can categorize tests according to features

With “Grouping” test concept, the integration test possibility is unlimited. For example, we can only test the “DatabaseFuntion” group from all of the unit test classes.

## 6. Parameterized Test

The “Parameterized Test” means vary parameter value for unit test. This feature is implemented in both JUnit 4 and TestNG. However both are using very different method to implement it.

JUnit 4

The “@RunWith” and “@Parameter” is use to provide parameter value for unit test, @Parameters have to return List[], and the parameter will pass into class constructor as argument.

It has many limitations here; we have to follow the “JUnit” way to declare the parameter, and the parameter has to pass into constructor in order to initialize the class member as parameter value for testing. The return type of parameter class is “List []”, data has been limited to String or a primitive value for testing.

TestNG

XML file or “@DataProvider” is use to provide vary parameter for testing.

**XML file for parameterized test.**  
Only “@Parameters” declares in method which needs parameter for testing, the parametric data will provide in TestNG’s XML configuration files. By doing this, we can reuse a single test case with different data sets and even get different results. In addition, even end user, QA or QE can provide their own data in XML file for testing.

**@DataProvider for parameterized test.**

While pulling data values into an XML file can be quite handy, tests occasionally require complex types, which can’t be represented as a String or a primitive value. TestNG handles this scenario with its @DataProvider annotation, which facilitates the mapping of complex parameter types to a test method.

TestNG’s parameterized test is very user friendly and flexible (either in XML file or inside the class). It can support many complex data type as parameter value and the possibility is unlimited

## 7. Dependency Test

The “Parameterized Test” means methods are test base on dependency, which will execute before a desired method. If the dependent method fails, then all subsequent tests will be skipped, not marked as failed.

JUnit 4

JUnit framework is focus on test isolation; it did not support this feature at the moment.

TestNG

It use “dependOnMethods “ to implement the dependency testing as following

## Conclusion

After go thought all the features comparison, i suggest to use TestNG as core unit test framework for Java project, because TestNG is more advance in parameterize testing, dependency testing and suite testing (Grouping concept). TestNG is meant for high-level testing and complex integration test. Its flexibility is especially useful with large test suites. In addition, TestNG also cover the entire core JUnit4 functionality. It’s just no reason for me to use JUnit anymore.