Comparative Analysis of JUnit and TestNG framework

Manasi Patil1, Mona Deshmukh2

Student, Dept. of MCA, VES Institute of Technology, Maharashtra, India1
Professor, Dept. of MCA, VES Institute of Technology, Maharashtra, India2

Abstract - Testing is an important phase in SDLC, Testing can be manual or automated. Nowadays Automation testing is widely used to find defects and to ensure the correctness and completeness of the software. Open source framework can be used for automation testing such as Robotframework, Junit, Spock, NUnit, TestNG, Jasmin, Mocha etc. This paper compares JUnit and TestNG based on their features and functionalities.

Key Words: JUnit, TestNG, Automation Framework, Automation Testing, Selenium TestNG, Selenium JUnit

1. INTRODUCTION

Testing a software manually is tedious work, and takes a lot of time and efforts. Automation saves lot of time and money, also it increases the test coverage and improves accuracy. It helps developers as well as testers. Choosing the right automation framework is crucial, so that it helps different kind of testing such as unit, functional, regression, volume or data driven testing. While choosing the right framework, it is important to take care of some factors such as framework should be reusable, and code should be maintainable, proper organisation and readability of the code, time optimization in terms of writing Tests and framework should have reporting mechanism. All frameworks are different and covers different key factors which are compared below.

1.1 Framework

Framework is a collection of rules or guidelines to interact with components which are interlinked. These components are responsible for the creation and execution of automated Tests and reporting for results after Test execution.

2. JUnit

It is popular open source java based light weight framework designed by Kent Beck, Erich Gamma and it is hosted at Github. It is test driven development (TDD) used for writing and running Tests. It uses annotations for identifying Tests, which are written as methods. It is a part of XUnit, Unit testing family. The current version used is Junit 5.

3. TestNG

It is an open source framework where NG stands for Next Generation in TestNG. This framework is inspired by JUnit and NUnit, but it overcomes the JUnit flaws. TestNG is user friendly and it covers unit, integration and functional testing. It is flexible than Junit and supports parametrization, parallel execution and data driven testing.

The Table – 1 compares different functionalities of TestNG and JUnit framework.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>TestNG</th>
<th>JUnit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Annotations</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Support Test Suit Initialization</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Support for Tests groups</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Support Parametrized Test</td>
<td>Yes</td>
<td>Limited support</td>
</tr>
<tr>
<td>Support Exception Test</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Support Dependency Test</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Support Ignore or Disable Test</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Support Parallel Testing</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Support Maven</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Support Data driven approach or dynamic input</td>
<td>Yes with Data provider</td>
<td>No</td>
</tr>
<tr>
<td>Support for Tests Reports</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IDE support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

4. Annotations

In simple language annotations are metadata for the code, i.e. data about the data. Annotations starts with ‘@’ followed by annotation name. They are introduced java JDK 5.0 onwards. They are applied before the method. Compiler checks for annotations and consider the following method as overridden method. Compiler checks for existence of the same method in the parent class otherwise throws a compiler error. Annotations does not directly affect the program execution but provide compile time instructions to the compiler.

The Table – 2 gives the list of different annotations of TestNG and JUnit framework for similar functionalities.
Table - 2: Annotations Support TestNG vs JUnit

<table>
<thead>
<tr>
<th>Annotations</th>
<th>TestNG Annotations</th>
<th>JUnit Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>For test annotations</td>
<td>@Test</td>
<td>@Test</td>
</tr>
<tr>
<td>Applied to methods to run before all tests in a Test Suite</td>
<td>@BeforeSuite</td>
<td></td>
</tr>
<tr>
<td>Applied to methods to run after all tests in a Test Suite</td>
<td>@AfterSuite</td>
<td>@BeforeSuite</td>
</tr>
<tr>
<td>Applied to methods to run before the test</td>
<td>@BeforeTest</td>
<td></td>
</tr>
<tr>
<td>Applied to methods to run after the test</td>
<td>@AfterTest</td>
<td></td>
</tr>
<tr>
<td>Applied to methods to run before the first Tests from the same group</td>
<td>@BeforeGroups</td>
<td></td>
</tr>
<tr>
<td>Applied to methods to run after the first Tests from the same group</td>
<td>@AfterGroups</td>
<td></td>
</tr>
<tr>
<td>Applied to methods to run before the first method in a class is invoked</td>
<td>@BeforeClass</td>
<td>@BeforeClass</td>
</tr>
<tr>
<td>Applied to methods to run after the first method in a class is invoked</td>
<td>@AfterClass</td>
<td>@AfterClass</td>
</tr>
<tr>
<td>Applied to the methods to run before each test method</td>
<td>@BeforeMethod</td>
<td>@Before</td>
</tr>
<tr>
<td>Applied to the methods to run after each test method</td>
<td>@AfterMethod</td>
<td>@After</td>
</tr>
<tr>
<td>To ignore a test method</td>
<td>@Ignore</td>
<td>@Test(enabled = false)</td>
</tr>
<tr>
<td>Timeout for test method</td>
<td>@Test(timeout = 1000)</td>
<td>@Test(timeout = 1000)</td>
</tr>
</tbody>
</table>

5. Annotations Syntax

Following are JUnit code snippets for above annotations:

```java
public class UnitAnnotations {
    
    @BeforeClass
    public static void BeforeClass_Method() {
        System.out.println("Annotation @BeforeClass - before all test cases");
    }

    @Before
    public void Before_Method() {
        System.out.println("Annotation @Before - before each test cases");
    }

    @After
    public static void AfterClass_Method() {
        System.out.println("Annotation @AfterClass - after all test cases");
    }

    @After
    public void After_Method() {
        System.out.println("Annotation @After - after each test cases");
    }

    @Test
    public void Test_Method(int a, int b) {
        int add = a + b;
        assertEquals(add, 10, "Assertion is correct");
    }

    @Ignore
    public void Ignore_Method() {
        System.out.println("Annotation @Ignore - this method is ignored");
    }

    @Test(timeout = 1000)
    public void Timeout_Method() {
        System.out.println("Annotation @Test(timeout) - enforce timeout in test case");
    }

    @Test(expected = ArithmeticException.class)
    public void Expected_Method() {
        int i = 10/0;
        System.out.println("Annotation @Test(expected) - test for specified exception during its execution");
    }
}
```

Following are TestNG code snippets for above annotations:

```java
public class TestNGAnnotations {
    
    @BeforeGroups("Regression")
    public void BeforeGroups_Method() {
        System.out.println("Annotation @BeforeGroups - before test executed from group");
    }

    @BeforeGroups("Regression")
    public void AfterGroups_Method() {
        System.out.println("Annotation @AfterGroups - after test executed from group");
    }

    @BeforeClass
    public void BeforeClass_Method() {
        System.out.println("Annotation @BeforeClass - before all test cases");
    }

    @AfterClass
    public static void AfterClass_Method() {
        System.out.println("Annotation @AfterClass - after all test cases");
    }

    @BeforeMethod
    public void Before_Method() {
        System.out.println("Annotation @Before - before each test cases");
    }

    @AfterMethod
    public void After_Method() {
        System.out.println("Annotation @After - after each test cases");
    }

    @Test(groups = "Regression")
    public void Test_Method1() {
        System.out.println("Annotation in the Regression group");
    }

    @Test
    public void Test_Method2() {
        System.out.println("Annotation @Test - Not included in any group");
    }

    @Test(timeout = 1000)
    public void Timeout_Method() {
        System.out.println("Annotation @Test(timeout) - enforce timeout in test case");
    }

    @Test(enabled = false)
    public void Ignore_Method() {
        System.out.println("Annotation @Ignore - This method is ignored");
    }

    @Test(expected = ArithmeticException.class)
    public void Expected_Method() {
        int i = 10/0;
    }
}
```
3. CONCLUSIONS

As per the above study both the frameworks are good for automation testing. Though JUnit is most popular and widely used testing framework, it does not offer much functionalities as compared to TestNG such as grouping, parallelism, dataProvider/parametrization, reporting. TestNG provides better flexibility than JUnit.

REFERENCES


