IEP-30: Migration to JUnit 5

**Motivation**
- Check to throw exceptions
- Ability to check timeouts
- Compatibility with Java 8
- Assumptions
- Useful to check flaky tests
- Parameterized Test
- Ability to use nested tests.
- Jupiter's extensions model.
- Test Annotations in interfaces.
- Condition test execution.
- Dependency injection for constructors and methods.
- Migration Process.

**Description**
- Process Overview
- Completion criteria for each point:
- Completion criteria on the whole:
- Additional improvements in JUnit 3-4 migration.

**Risks and Assumptions**

**Discussion Links**

**Reference Links**

**Tickets**

**Motivation**

New useful functionality will be added:

- **Check to throw exceptions**

```java
public void shouldRaiseAnException () throws Exception {
    Assertions.assertThrows(Exception.class, () -> {
        //...
    });
}
```

In JUnit 4:

```java
@Test(expected = Exception.class)
```

- **Ability to check timeouts**
@Test(timeout = 1)
public void shouldFailBecauseTimeout () throws InterruptedException {
    Thread.sleep(10);
}

@Test
public void shouldFailBecauseTimeout () throws InterruptedException {
    Assertions.assertTimeout(Duration.ofMillis(1), () -> Thread.sleep(10));
}

• Compatibility with Java8

Now it is possible to write lambdas directly from Assert:

@Test
public void shouldFailBecauseTheNumbersAreNotEqual_lazyEvaluation () {
    Assertions.assertTrue(
        2 == 3,
        () -> "Numbers " + 2 + " and " + 3 + " are not equal!");
}

@Test
public void shouldAssertAllTheGroup () {
    List<Integer> list = Arrays.asList(1, 2, 4);
    Assertions.assertAll("List is not incremental",
        () -> Assertions.assertEquals(list.get(0).intValue(), 1),
        () -> Assertions.assertEquals(list.get(1).intValue(), 2),
        () -> Assertions.assertEquals(list.get(2).intValue(), 3));
}

• Assumptions

Tests will be executed under conditions:

@Test
public void whenEnvironmentIsWeb_thenUrlsShouldStartWithHttp () {
    assumingThat("WEB”.equals(System.getenv(“ENV”)),
        () -> {
            assertTrue("http".startsWith(address));
        });
}

• Useful to check flaky tests

@RepeatedTest(value = 3, name = "Custom name {currentRepetition}/{totalRepetitions}"
void repeatedTestWithCustomDisplayName (TestInfo testInfo){
    assertEquals(2, Math.addExact(1, 1), "1 + 1 should equal 2");
}

• Parameterized Test
@ParameterizedTest
@ValueSource(strings = {"Hello", "JUnit"})
void withValueSource (String word){
    assertNotNull(word);
}


• Ability to use nested tests.

Can contain one @BeforeEach method, and one @AfterEach method, but because Java doesn’t allow static members in inner classes, the @BeforeAll and @AfterAll methods don’t work by default.

class JUnit5NestedExampleTest {

    ...

    @Nested
    @DisplayName("Tests in the nested class A") class A {

        ...

        @Test
        @DisplayName("Example test for method A")
        void sampleTestForMethodA() {
            //test logic
        }

        @Nested
        @DisplayName("Another nested test class") class B {
            //test logic
        }
    }
}

• Jupiter's extensions model.

Extensions model allows third parties to extend JUnit with their own additions. It is useful when a developer wants to use some features that absent in default JUnit version.

Now five main types of extension points can be used: test instance post-processing, conditional test execution, life-cycle callback, parameter resolution, exception handling.

@ExtendWith(BenchmarkExtension.class)
public @interface Benchmark { ... }

public class BenchmarkExtension implements
    BeforeAllCallback, BeforeTestExecutionCallback,
    AfterTestExecutionCallback, AfterAllCallback { /* extension logic */}

An extension will be executed only if a class marked the corresponding annotation. For detailed information, please take a look at the tutorial.

• Test Annotations in interfaces.

JUnit5 allows the next annotations in interfaces:

1. @Test, @TestFactory, @BeforeEach, @AfterEach can be added to default methods in interfaces
2. @BeforeAll and @AfterAll can be added to static methods in interfaces
3. @ExtendsWith and @Tag can be declared on interfaces

And the classes that implement these interface will inherit the test cases.
• **Condition test execution.**

5 version allows to define custom annotations that act as conditions to determine whether a test should be run or not. It is necessary to create annotation for condition

```java
@Target({ ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
@ExtendWith(DisabledOnEnvironmentCondition.class)
public @interface DisabledOnEnvironment {
    String[] value();
}
```

And class that implements the appropriate interface and override the evaluate() method

```java
@override
public ConditionEvaluationResult evaluate(TestExtensionContext context) {
    // evaluation logic
}
```

Now we can add annotation on the test method

```java
@Test
@DisabledOnEnvironment({"dev", "prod"})
void testFail() {
    fail("this test fails");
}
```

• **Dependency injection for constructors and methods.**

Now it becomes available to define parameters for test constructors and methods and use dependency injection for them. With ParameterResolver class we can resolve parameters in runtime.

At the time of this writing there are 3 built-in parameter resolvers: TestInfo, RepetitionInfo, TestReporter. Example for one of them:

```java
@Test
@DisplayName("Test Get Users")
public void testGetUsersNumberWithInfo(TestInfo testInfo) {
    // test logic.
    logger.info("Running test method:" + testInfo.getTestMethod().get().getName());
}
```

• **Migration Process.**

Migration from JUnit 4 to 5 is not so difficult, as backward compatibility is available.

**Description**

The new features of JUnit 5 can improve Apache Ignite development process and make testing more effectively. Moreover, JUnit 5 has totally backward compatibility with 4 version (it is achievable with JUpiter module).

**Process Overview**

1. Migrate examples-module.
   a. Add dependencies to parent/pom.xml
   b. Change adding suites to IgniteExamplesSelfTestSuite. Pay more attention for this point, because @RunWith(JUnitPlatform.class) construction can provide not fully support of all features
   c. Launch IgniteExamplesMLTestSuite separately under JUnit4, because it is impossible to override JUnitPlatform constructor as well as Suite constructor
   d. Change annotations and imports in test classes
   e. Ignore commented tests
2. Migrate core module in the same way as examples. Classes with dynamic tests do not need to be changed. This is due to the fact that in JUnit5 it is not possible to launch static and dynamic tests under one surefire version. Positive side: the dynamic tests amount is not so big, they locate in specific places. The all JUnit5 opportunity can not be used in dynamic tests (@TestFactory).

According to the previous points, the next should be added:
- Remove test timeouts and add timeouts via annotations
- Replace exception checking according to JUnit5 style
- Replace some tests on parameterized where it is necessary (the same tests are used with different parameters)

3. Migrate modules where JUnit tests exist. Check list:
- aop
- aws
- camel
- cassandra
- clients
- cloud
- compatibility
- compress
- direct-io
- flink
- flume
- gce
- geospastial
- hadoop
- hibernate 4.2, 5.1, 5.3, core
- ignored
- indexing
- jcl
- jta
- kafka
- kubernetes
- log4j
- log4j2
- ml
- mqtt
- osgi
- rest-http
- rocketmq
- schedule
- slf4j
- spark
- spring, spring-data, 2.0
- ssh
- storm
- tensorflow
- twitter
- urldeploy
- web-console
- yarn
- zeromq
- zookeeper

Completion criteria for each point:

Tests can be launched under JUnit5. Usage of JUnit 5 annotations becomes available, for example, @RepeatedTest

There are no additional fails on TeamCity.

Completion criteria on the whole:

All modules satisfy the previous two criteria

The community is informed about the migration to 5 and benefits of the newest version.

Additional improvements in JUnit 3->4 migration.

Actualize JUnit3TestLegacyAssert class: rename it, update documentation. Replacement inheritance to imports is not appropriate here as it leads to changes of all test classes. Moreover, during migration to the next JUnit versions, problem with changing all test classes will arise again and in new test classes developer should make import manually.

In JUnit3TestLegacySupport:

1. Investigate replacement of beforeTest/afterTest and beforeTestsStarted/afterTestsStopped methods and if needed replace them by annotations - it is noticed in IGNITE-11413.
2. Replace tearDown, setUp by beforeTest, afterTest methods where it is necessary.
3. Actualize JUnit3TestLegacySupport: remove deprecated annotation, rename class and specify documentation.
Futhermore, GridAbstractTest and GridCommonAbstractTest refactoring necessity should be investigated. Check ability to turn some instance methods to static methods. It was mentioned in the conversation on dev-list.

**Risks and Assumptions**

- Additional fails on TC because of migration issues
- Performance degradation of tests execution.

**Discussion Links**

Devlist discussion.

**Reference Links**

*Question* to JUnit develop team about surefire version.

**Tickets**

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>T</th>
<th>Created</th>
<th>Updated</th>
<th>Due</th>
<th>Assignee</th>
<th>Reporter</th>
<th>P</th>
<th>Status</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGNITE-11414</td>
<td>Remove JUnit3LegacySupport</td>
<td>↑</td>
<td>Feb 25, 2019</td>
<td>Apr 03, 2019</td>
<td></td>
<td>Ivan Fedotov</td>
<td>Ivan Fedotov</td>
<td></td>
<td>CLOSED</td>
<td>Not A Problem</td>
</tr>
<tr>
<td>IGNITE-11413</td>
<td>Remove beforeTestsStarted, afterTestsStarted from JUnit3TestLegacySupport</td>
<td>↑</td>
<td>Feb 25, 2019</td>
<td>Apr 03, 2019</td>
<td></td>
<td>Ivan Fedotov</td>
<td>Ivan Fedotov</td>
<td></td>
<td>CLOSED</td>
<td>Not A Problem</td>
</tr>
<tr>
<td>IGNITE-11708</td>
<td>Unable to run tests in IgniteConfigVariationsAbstractTest subclasses</td>
<td>⇩</td>
<td>Apr 09, 2019</td>
<td>Sep 12, 2019</td>
<td></td>
<td>Ivan Fedotov</td>
<td>Ivan Fedotov</td>
<td></td>
<td>RESOLVED</td>
<td>Fixed</td>
</tr>
<tr>
<td>IGNITE-10180</td>
<td>Investigate migration from Junit 4 to 5</td>
<td>↑</td>
<td>Nov 07, 2018</td>
<td>Feb 21, 2019</td>
<td></td>
<td>Ivan Fedotov</td>
<td>Oleg Ignatenko</td>
<td></td>
<td>RESOLVED</td>
<td>Done</td>
</tr>
<tr>
<td>IGNITE-11412</td>
<td>Actualize JUnit3TestLegacySupport class</td>
<td>↑</td>
<td>Feb 25, 2019</td>
<td>Apr 24, 2019</td>
<td></td>
<td>Ivan Fedotov</td>
<td>Ivan Fedotov</td>
<td></td>
<td>RESOLVED</td>
<td>Fixed</td>
</tr>
<tr>
<td>IGNITE-11411</td>
<td>Remove tearDown, setUp from JUnit3TestLegacySupport</td>
<td>↑</td>
<td>Feb 25, 2019</td>
<td>Apr 10, 2019</td>
<td></td>
<td>Ivan Fedotov</td>
<td>Ivan Fedotov</td>
<td></td>
<td>RESOLVED</td>
<td>Fixed</td>
</tr>
<tr>
<td>IGNITE-11396</td>
<td>Actualize JUnit3TestLegacyAssert</td>
<td>↑</td>
<td>Feb 22, 2019</td>
<td>Mar 18, 2019</td>
<td></td>
<td>Ivan Fedotov</td>
<td>Ivan Fedotov</td>
<td></td>
<td>RESOLVED</td>
<td>Fixed</td>
</tr>
<tr>
<td>IGNITE-10973</td>
<td>Migrate example module tests from Junit 4 to 5</td>
<td>↑</td>
<td>Jan 18, 2019</td>
<td>Aug 13, 2019</td>
<td></td>
<td>Ivan Fedotov</td>
<td>Ivan Fedotov</td>
<td></td>
<td>IN PROGRESS</td>
<td>Unresolved</td>
</tr>
<tr>
<td>IGNITE-10958</td>
<td>Migrate from Junit 4 to 5</td>
<td>✓</td>
<td>Jan 16, 2019</td>
<td>Oct 10, 2019</td>
<td></td>
<td>Ivan Fedotov</td>
<td>Ivan Fedotov</td>
<td></td>
<td>OPEN</td>
<td>Unresolved</td>
</tr>
</tbody>
</table>

9 issues