Migration to JUnit 5 in the legacy project

14 July 2018

About me

Artem Vegera

8 years work experience

Senior Java Developer (Exadel)

Goals

- JUnit 5 architecture overview
- JUnit 5 new features
- Pragmatic migration from JUnit 4 to 5

0. JUnit Status

?

JUnit 4 Releases

4.0 (released 16 Feb, 2006)

...

4.12 (released 4 Dec, 2014)

4.13 (unreleased)

?

JUnit 4 Problems

- Parameterized tests out of the box
- Extension mechanism
- Only one @RunWith per test class
- Not easy implementation of new runner
- Doesn't use Java 8+ features
- Other test frameworks evolved

JUnit 5 Crowdfunding

INDIEGOGO

CLOSED

JUnit Lambda

Shaping the future of testing on the JVM

PROJECT OWNER

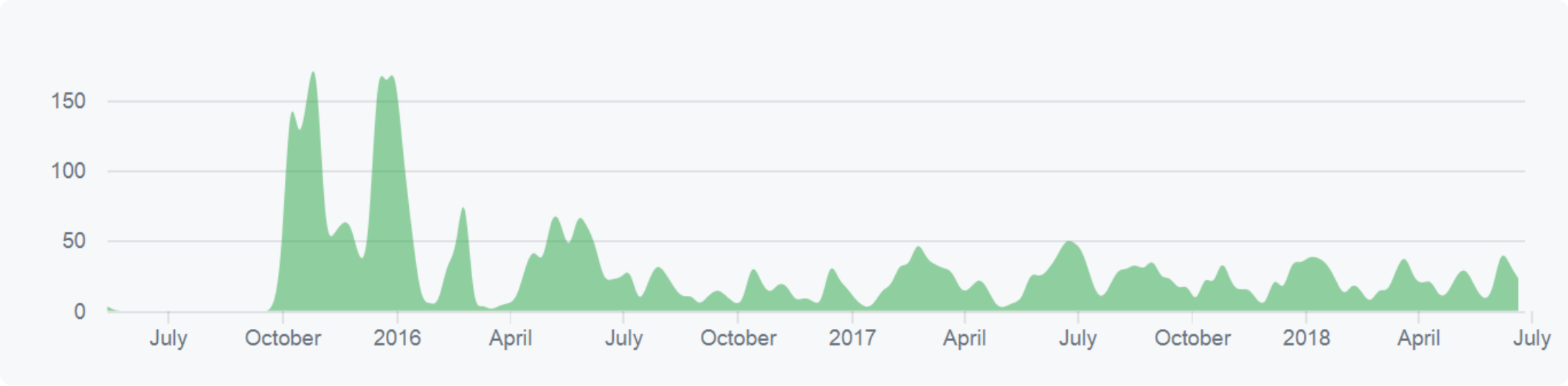
Marc Philipp
Karlsruhe, Germany
1 Campaign | More

€25,000 EUR fixed goal

Raised €53,937

JUnit 5 Contributors

May 31, 2015 – Jul 14, 2018



sbrannen #1
1,904 commits 117,689 ++ 82,799 --

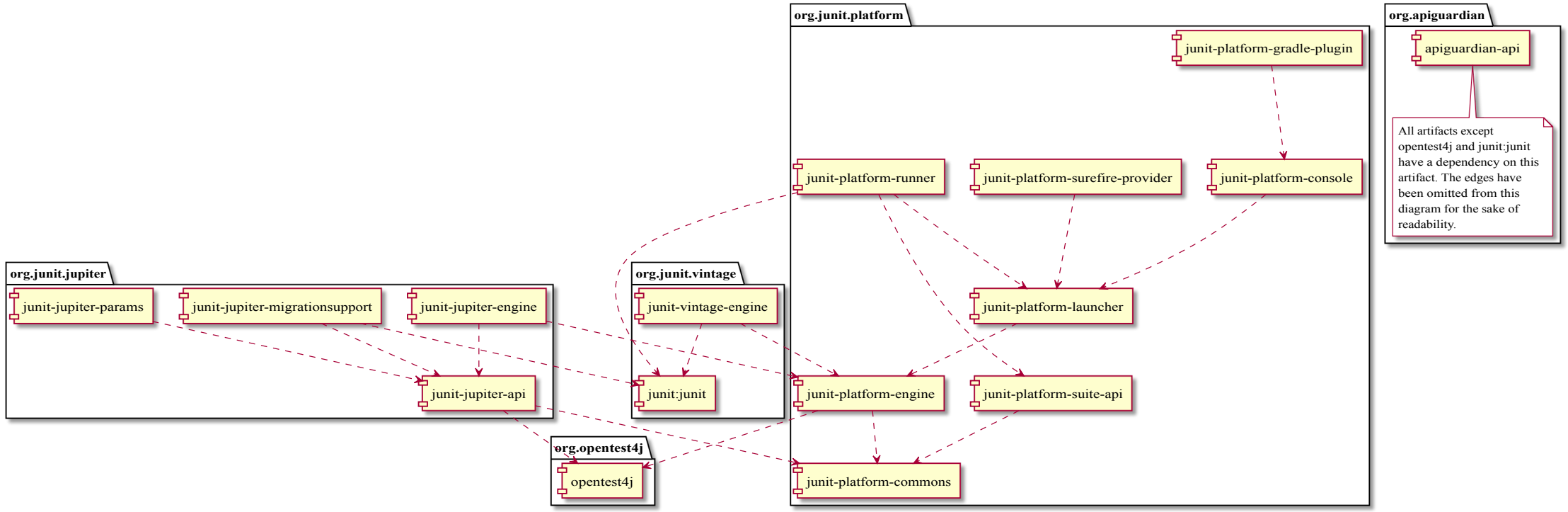
marcphilipp #2
1,142 commits 70,641 ++ 40,008 --

jlink #3
551 commits 35,009 ++ 23,434 --

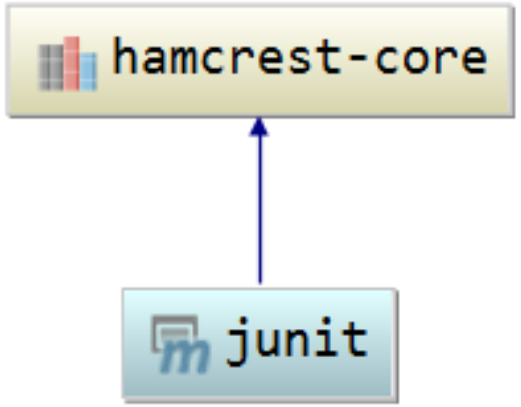
sormuras #4
407 commits 30,041 ++ 12,286 --

I. JUnit Platform overview

JUnit 5 Modules



JUnit 4 Modules



So, will this work?

```
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
-     <version>4.12</version>
+     <version>5.0</version>
      <scope>test</scope>
    </dependency>
```

No. Because JUnit 5

- is not single JAR
- It has 10+ jars in the 3 groups

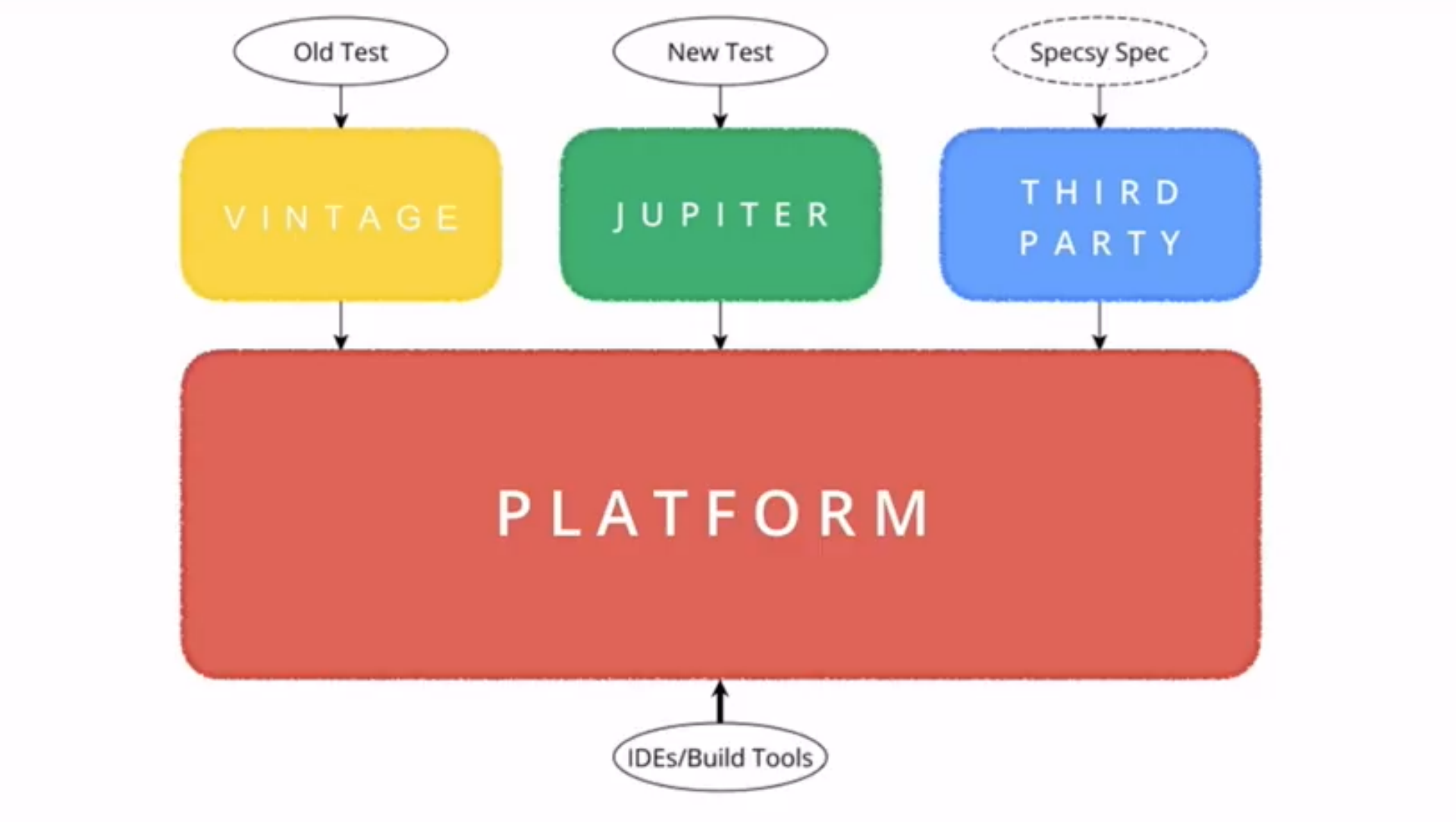
JUnit 5 Architecture

JUnit 5

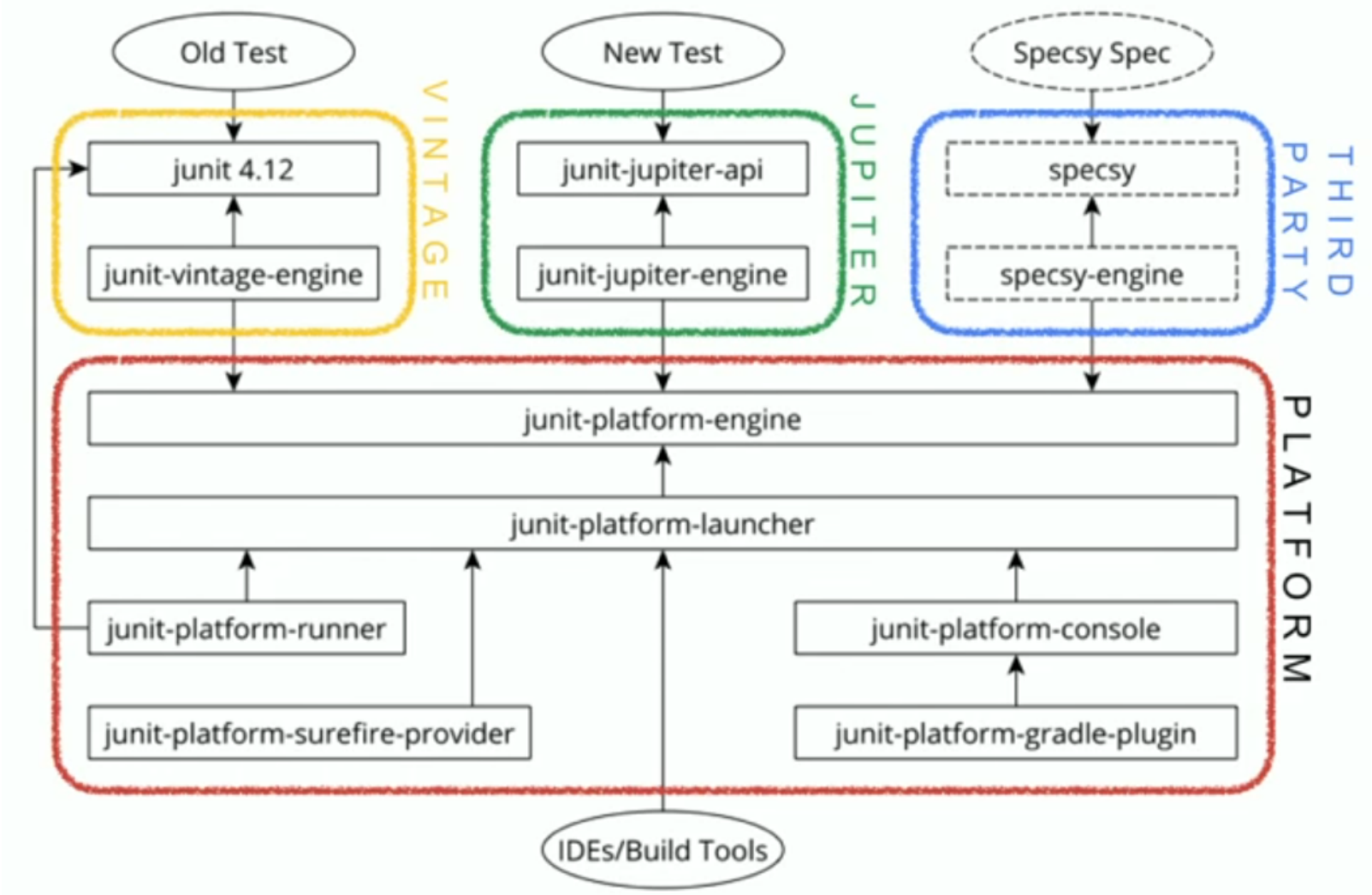
=

JUnit Platform + JUnit Jupiter + JUnit Vintage

JUnit Platform



JUnit Platform (detailed view)



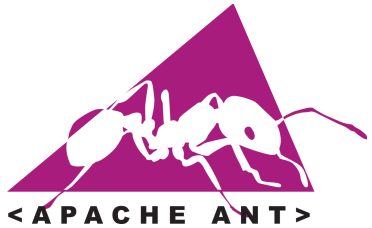
JUnit 5 Support

4.7.1a+

2016.2+

2.21.0+

4.6+



1.10.3+

Third-Party test Engines

- Specsy
- Spek
- KotlinTest
- Cucumber
- Drools Scenario
- jqwik

Open test Alliance for JVM

Test NG
Hamcrest
AssertJ
Spock
Google Truth
ScalaTest

Eclipse
IntelliJ
Gradle
Maven Surefire Plugin
Allure Framework

Migration from JUnit 4 to JUnit 5

Maven Migration Example

Dependency

```
<dependency>
    <groupId>org.junit.vintage</groupId>
    <artifactId>junit-vintage-engine</artifactId>
    <version>5.2.0</version>
    <scope>test</scope>
</dependency>
```

Maven Surefire

```
<plugin>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.22.0</version>
    <dependencies>
        <dependency>
            <groupId>org.junit.platform</groupId>
            <artifactId>junit-platform-surefire-provider</artifactId
            <version>1.2.0</version>
        </dependency>
    </dependencies>
</plugin>
```

JUnit 5 Platform (recap)

- JUnit 5 = Platform + Jupiter + Vintage
- Third-party test engines
- Migration cost is 1 PR with pom.xml / build.gradle file changes

II. JUnit Jupiter overview

First JUnit 5 Test

```
import static org.junit.jupiter.api.Assertions.assertEqu

import org.junit.jupiter.api.Test;

class FirstJUnit5Test {

    @Test
    void firstJUnit5Test() {
        assertEquals(2, 1 + 1);
    }
}
```

?

@Test

```
package org.junit;

@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD})
public @interface Test {

    static class None extends Throwable {
        private static final long serialVersionUID =

        private None() {
        }
    }

    Class<? extends Throwable> expected() default No

    long timeout() default 0L;
}
```

```
@Target({ ElementType.ANNOTATION_TYPE, ElementT
@Retention(RetentionPolicy.RUNTIME)
@Documented
@API(status = STABLE, since = "5.0")
@Testable
public @interface Test {
}
```

@BeforeAll and @AfterAll

```java
public class JUnit4BeforeClassAndAfterClassTest {

    @BeforeClass
    public static void beforeClass() {
        System.out.println("beforeClass()");
    }

    @Test
    public void multiplyTest1() {
        System.out.println("multiplyTest1()");
        assertEquals(40, 4 * 10);
    }

    @Test
    public void multiplyTest2() {
        System.out.println("multiplyTest2()");
        assertEquals(50, 5 * 10);
    }

    @AfterClass
    public static void afterClass() {
        System.out.println("afterClass()");
    }
}
```

```java
class JUnit5BeforeAllAndAfterAllTest {

    @BeforeAll
    static void beforeAll() {
        System.out.println("beforeAll()");
    }

    @Test
    void multiplyTest1() {
        System.out.println("multiplyTest1()");
        assertEquals(40, 4 * 10);
    }

    @Test
    void multiplyTest2() {
        System.out.println("multiplyTest2()");
        assertEquals(50, 5 * 10);
    }

    @AfterAll
    static void afterAll() {
        System.out.println("afterAll()");
    }
}
```

@BeforeEach and @AfterEach

```java
public class JUnit4BeforeAndAfterTest {

    @Before
    public void before() {
        System.out.println("before()");
    }

    @Test
    public void multiplyTest1() {
        System.out.println("multiplyTest1()");
        assertEquals(40, 4 * 10);
    }

    @Test
    public void multiplyTest2() {
        System.out.println("multiplyTest2()");
        assertEquals(50, 5 * 10);
    }

    @After
    public void after() {
        System.out.println("after()");
    }
}
```

```java
class JUnit5BeforeEachAndAfterEachTest {

    @BeforeEach
    void beforeEach() {
        System.out.println("beforeEach()");
    }

    @Test
    void multiplyTest1() {
        System.out.println("multiplyTest1()");
        assertEquals(40, 4 * 10);
    }

    @Test
    void multiplyTest2() {
        System.out.println("multiplyTest2()");
        assertEquals(50, 5 * 10);
    }

    @AfterEach
    void afterEach() {
        System.out.println("afterEach()");
    }
}
```

@Disabled

```java
public class JUnit4IgnoreTest {

    @Test
    @Ignore
    public void multiplyTest1() {
        assertEquals(4, 2 + 2);
    }
}
```

```java
class JUnit5DisabledTest {

    @Test
    @Disabled
    void multiplyTest1() {
        assertEquals(4, 2 + 2);
    }
}
```

Assertions and Assumptions

Looks similar, but a few new ones.

JUnit 4

```
import static org.junit.Assert.*;
import static org.junit.Assume.*;
```

JUnit 5

```
import static org.junit.jupiter.api.Assertions.*;
import static org.junit.jupiter.api.Assumptions.*;
```

assertThrows()

```java
public class JUnit4AssertThrowsTest {

    @Rule
    public ExpectedException exceptionRule = ExpectedExceptic

    @Test
    public void testExceptionThrowingUsingRule() {
        exceptionRule.expect(NumberFormatException.class);
        exceptionRule.expectMessage("For input string");
        Integer.parseInt("1a");
    }

    @Test(expected = NullPointerException.class)
    public void testExceptionThrowing() {
        String test = null;
        test.length();
    }
}
```

```java
class JUnit5AssertTrowsTest {

    @Test
    void assertThrowsExample() {
        String test = null;
        assertThrows(NullPointerException.class, () -> test.
    }
}
```

assertTimeout()

```
public class JUnit4AssertTimeoutTest {

    @Test(timeout = 100)
    public void testTimeout() throws InterruptedException {
        Thread.sleep(10);
    }
}
```

```
class JUnit5AssertTimeoutTest {

    @Test
    void shouldFailBecauseTimeout() {
        Assertions.assertTimeout(Duration.ofMillis(100), () -> Thread.sleep(10));
    }
}
```

@Tag

```
public class JUnit4CategoryTest {

    @Test
    @Category(IntegrationTest.class)
    public void testReadingFile() throws IOException {
        String content = new String(Files.readAllBytes(Paths.get("testFile.txt")));
    }
}
```

```
class JUnit5TagTest {

    @Test
    @Tag("IntegrationTest")
    void testTag() throws IOException {
        String content = new String(Files.readAllBytes(Paths.get("testFile.txt")));
    }
}
```

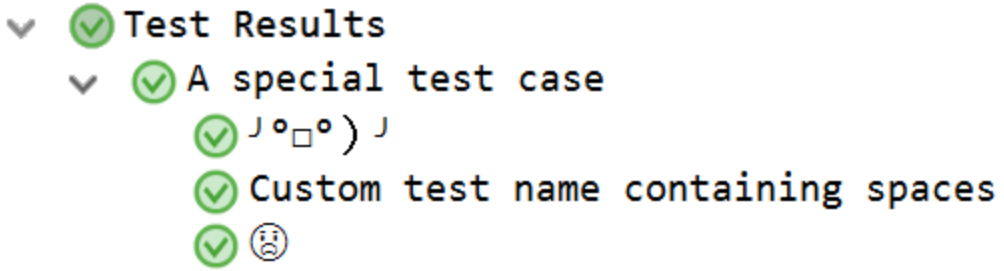
Same Features Recap

JUnit 5	JUnit 4
@Test	@Test
@BeforeEach	@Before
@BeforeAll	@BeforeClass
@AfterEach	@After
@AfterAll	@AfterClass
@Disabled	@Ignore
@Tag	@Category
Assertions (extended)	Assertions
Assumptions (extended)	Assumptions

III. JUnit 5 new features

@DisplayName

```java
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

@DisplayName("A special test case")
class DisplayNameDemo {

    @Test
    @DisplayName("Custom test name containing spaces")
    void testWithDisplayNameContainingSpaces() {
    }

    @Test
    @DisplayName("╯°□°）╯")
    void testWithDisplayNameContainingSpecialCharacters() {
    }

    @Test
    @DisplayName("😱")
    void testWithDisplayNameContainingEmoji() {
    }

}
```

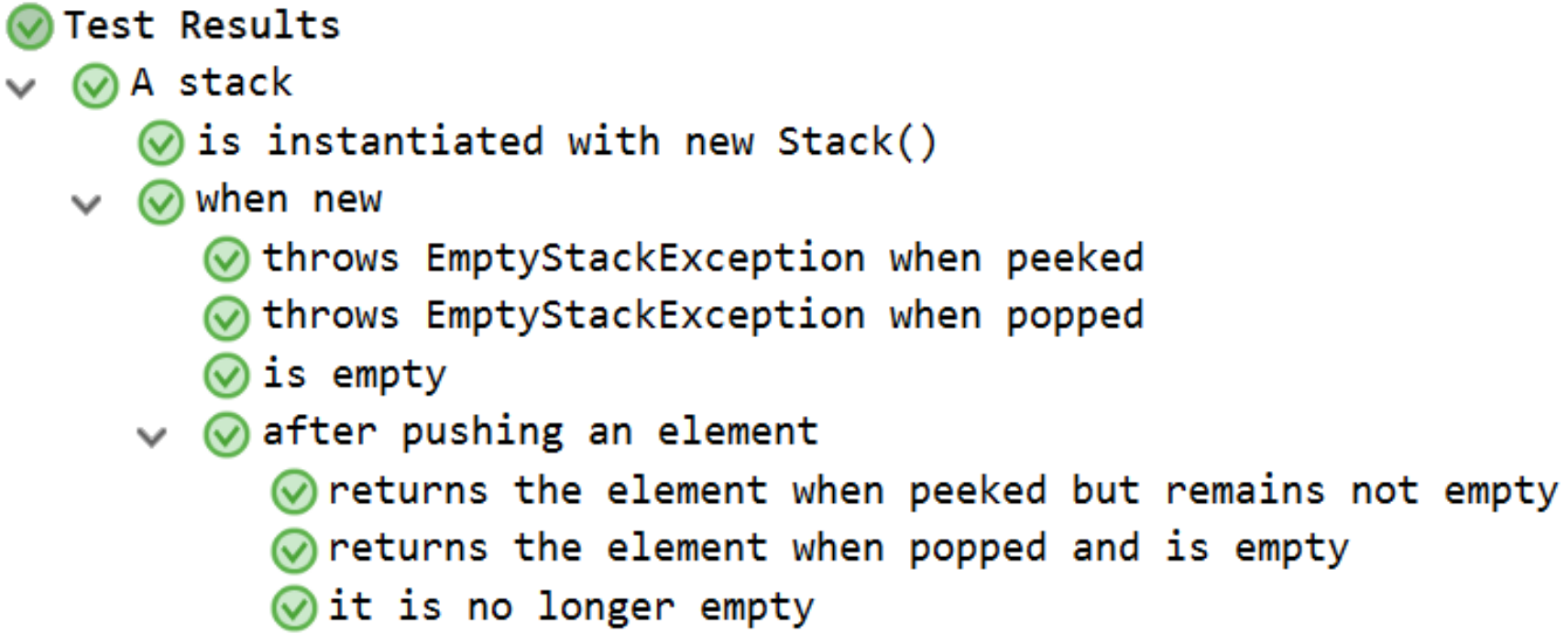


Nested Tests

```
@DisplayName("A stack")
class JUnit5NestedTest {

    private Stack<Object> stack;

    @Test
    @DisplayName("is instantiated with new Stack
    void isInstantiatedWithNew() {
        new Stack<>();
    }

    @Nested
    @DisplayName("when new")
    class WhenNew {

        @BeforeEach
        void createNewStack() {
            stack = new Stack<>();
        }

        @Test
        @DisplayName("is empty")
        void isEmpty() {
            assertTrue(stack.isEmpty());
        }
```

```
    @Test
    @DisplayName("throws EmptyStackException when popped")
    void throwsExceptionWhenPopped() {
        assertThrows(EmptyStackException.class, () -> stack.pop
    }

    ...

    @Nested
    @DisplayName("after pushing an element")
    class AfterPushing {

        String anElement = "an element";

        @BeforeEach
        void pushAnElement() {
            stack.push(anElement);
        }

        @Test
        @DisplayName("it is no longer empty")
        void isNotEmpty() {
            assertFalse(stack.isEmpty());
        }
        ...
    }
```

Nested Tests Execution



@ExtendWith

@Rule + @RunWith

(composable + powerful)

=

@ExtendWith

(composable & powerful)

@ExtendWith

Class level

```
@ExtendWith(MockitoExtension.class)
public class JUnit5MockitoExtensionTest {
```

Method level

```
@ExtendWith(MockitoExtension.class)
@Test
void mockTest() {
    // ...
}
```

Multiple extensions

```
@ExtendWith({ FooExtension.class, BarExtension.class })
class MyTestsV1 {
    // ...
}
```

Extention points

- TestInstancePostProcessor
- ParameterResolver
- ContainerExecutionCondition
- TestExecutionCondition
- TestExecutionExceptionHandler
- BeforeAllCallback
- BeforeEachCallback
- BeforeTestExecutionCallback
- AfterTestExecutionCallback
- AfterEachCallback
- AfterAllCallback

Third-Party Extensions

https://github.com/junit-team/junit5/wiki/Third-party-Extensions

- SpringExtension
- MockitoExtension
- Docker
- upREST
- JGiven
- SBT Jupiter Interface
- JPA Unit
- selenium-jupiter
- jFairy
- Database Rider
- Rerunner-Jupiter
- Wiremock
- mockito-rest-spring
- Kafka JUnit
- JUnit Extensions
- JUnit Pioneer
- Jersey JUnit
- Unroll Extension
- GreenMail
- S3Mock
- Citrus Framework
- XWiki Component Mocking

Third-Party Extensions

https://github.com/junit-team/junit5/wiki/Third-party-Extensions

```
@ExtendWith(MockitoExtension.class)
class MockitoDemoTest {

    @BeforeEach
    void init(@Mock Person person) {
        when(person.getName()).thenReturn("Dilbert");
    }

    @Test
    void simpleTestWithInjectedMock(@Mock Person person) {
        assertEquals("Dilbert", person.getName());
    }

}
```

Dependency Injection

```
@DisplayName("TestInfo Demo")
class TestInfoDemo {

    TestInfoDemo(TestInfo testInfo) {
        assertEquals("TestInfo Demo", testInfo.getDisplayName());
    }

    @BeforeEach
    void init(TestInfo testInfo) {
        String displayName = testInfo.getDisplayName();
        assertTrue(displayName.equals("TEST 1") || displayName.equals("test2()"));
    }

    @Test
    @DisplayName("TEST 1")
    @Tag("my-tag")
    void test1(TestInfo testInfo) {
        assertEquals("TEST 1", testInfo.getDisplayName());
        assertTrue(testInfo.getTags().contains("my-tag"));
    }

}
```

Dynamic Tests

```
class DynamicTestsDemo {

    @TestFactory
    Collection<DynamicTest> dynamicTestsFromCollection() {
        return Arrays.asList(
            dynamicTest("1st dynamic test", () -> assertTrue(true)),
            dynamicTest("2nd dynamic test", () -> assertEquals(4, 2 * 2))
        );
    }

    @TestFactory
    Stream<DynamicTest> dynamicTestsFromIntStream() {
        // Generates tests for the first 10 even integers.
        return IntStream.iterate(0, n -> n + 2)
                .limit(10)
                .mapToObj(n ->
                    dynamicTest("test" + n, () -> assertTrue(n % 2 == 0))
                );
    }
}
```

Parametrized Tests

New dependency

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-params</artifactId>
    <version>5.2.0</version>
    <scope>test</scope>
</dependency>
```

```
@ParameterizedTest
@ValueSource(strings = { "Hello", "World" })
void testWithStringParameter(String argument) {
    assertNotNull(argument);
}
```

Parametrized tests with different source

@EnumSource

@MethodSource

@ValueSource

@CsvSource

@CsvFileSource

...

@RepeatedTest

```
@RepeatedTest(10)
void repeatedTest() {
    // ...
}
```

```
@RepeatedTest(value = 10, name = "{displayName}[currentRepetition]")
```

Placeholders:

{displayName}

{currentRepetition}

{totalRepetitions}

@TestInstance

```
//Custom
@TestInstance(Lifecycle.PER_CLASS)
class DynamicTestsDemo {

    ...
}

//By default
@TestInstance(Lifecycle.PER_METHOD)
class DynamicTestsDemo {

    ...
}
```

Conditional Test Execution

```
@EnabledOnOs(MAC)

@EnabledOnJre(JAVA_8)

@EnabledIfSystemProperty(named = "os.arch", matches = ".*64.*")

// Dynamic JavaScript expression.
@DisabledIf("Math.random() < 0.314159")
void mightNotBeExecuted() {
    ...
}
```

New features recap

- @DisplayName
- @Nested
- @ExtendWith (with 3rd-party and your own extensions)
- Dependency Injection
- Dynamic Tests
- Parameterized Tests with many possible sources
- Repeated Tests
- @TestInstance
- Conditional Test Execution

IV. Migration of test code from JUnit 4 to 5

Migration Guide

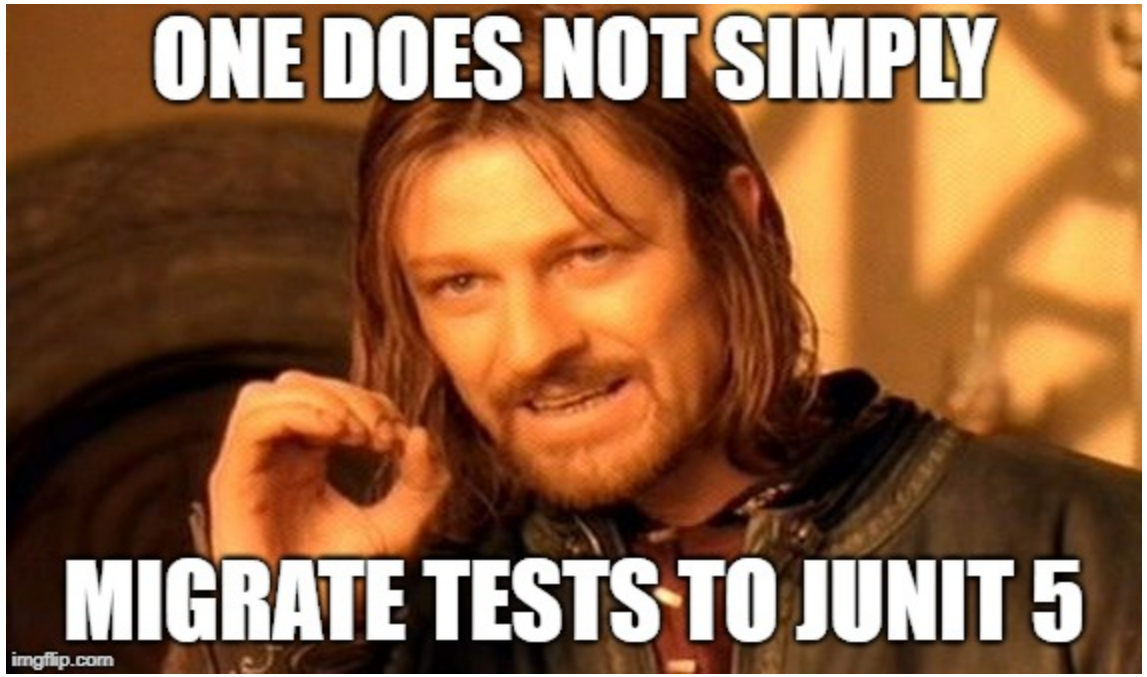
- The following are things you have to watch out for when migrating existing JUnit 4 tests to JUnit Jupiter.
- Annotations reside in the org.junit.jupiter.api package.
- Assertions reside in org.junit.jupiter.api.Assertions.
- Assumptions reside in org.junit.jupiter.api.Assumptions.
- @Before and @After no longer exist; use @BeforeEach and @AfterEach instead.
- @BeforeClass and @AfterClass no longer exist; use @BeforeAll and @AfterAll instead.
- @Ignore no longer exists: use @Disabled instead.
- @Category no longer exists; use @Tag instead.
- @RunWith no longer exists; superseded by @ExtendWith.
- @Rule and @ClassRule no longer exist; superseded by ExtendWith; see the following section for partial rule support.

Migration Tip #1

Use the new API and all the new features for new test classes.

Migration Tip #2

Don’t try to migrate all existing tests. It causes a lot of effort with no direct business value.



Migration Tip #3

Instead, apply the boyscout rule by gradually migrating existing tests before they need to be changed.

Migration Tip #4

One way to get accustomed to the new API is to migrate some (not all) existing tests, preferably the most complex ones.

Conclusion

Follow up

So, today we got answers:

1. What is JUnit Platform?
2. How to migrate your project from JUnit 4 to 5?
3. Hot to use JUnit 5 new features?
4. How to migrate your tests from JUnit 4 to 5

Thanks for attention

User Guide https://junit.org/junit5/docs/current/user-guide/
Samples https://github.com/junit-team/junit5-samples

Slides https://slides.com/atomicus/junit5-migration
Code https://github.com/avegera/junit5-talk

Q & A