



adaptTo()

APACHE SLING & FRIENDS TECH MEETUP
10-12 SEPTEMBER 2018

JUnit 5 and Sling/AEM Mocks
Stefan Seifert, pro!vision GmbH

About the Speaker

- AEM Developer
- Apache Sling PMC
- CTO of pro!vision GmbH



Stefan Seifert



PRO!VISION
SOFTWARE CRAFTSMANSHIP

<https://www.pro-vision.de>

JUnit 5 – Top 10 New Features

JUnit The JUnit logo consists of the word "JUnit" in a grey, sans-serif font, followed by a circular icon. The icon is split vertically: the left half is red and the right half is green. A white number "5" is centered within the circle.

#1 Visibility – Less boilerplate code

No more need
to make classes,
methods public

```
class visibilityTest {  
  
    private MyClass underTest;  
  
    @BeforeEach  
    void setUp() {  
        underTest = new MyClass();  
    }  
  
    @Test  
    void testTheAnswer() {  
        assertEquals(42, underTest.theAnswer(),  
            "Answer to the Ultimate Question of Life, "  
            + "the Universe, and Everything");  
    }  
}
```

#2 Display names

```
@DisplayName("Display name demonstration")
class DisplayNameTest {

    @Test
    @DisplayName("This tests the correct answer")
    void testTheAnswer() {
        assertEquals(42, new MyClass().theAnswer());
    }

    @Test
    @DisplayName("This get's the answer wrong")
    void testTheWrongAnswer() {
        assertEquals(45, new MyClass().theAnswer());
    }
}
```

Give classes and tests nice names

Finished after 1,22 seconds

Runs: 5/5

Errors: 0

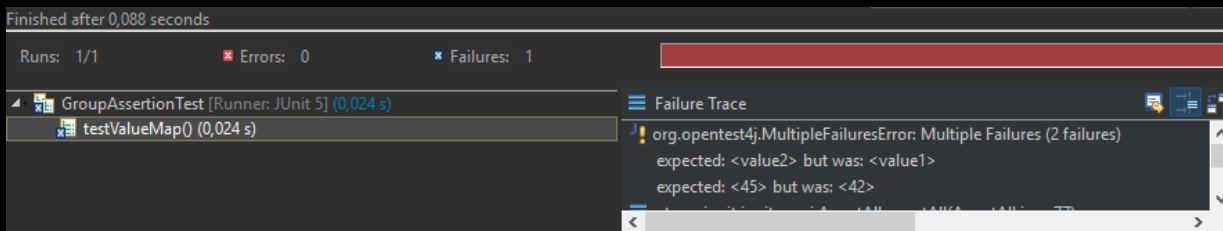
Failures: 1

```
Display name demonstration [Runner: JUnit 5] (0,000 s)
  This tests the correct answer (0,000 s)
  This get's the answer wrong (0,000 s)
```

#3 Group assertions

```
class GroupAssertionTest {  
  
    @Test  
    void testValueMap() {  
        ValueMap props = new MyClass().getProps();  
        assertAll(  
            () -> assertEquals("value2", props.get("prop1", "")),  
            () -> assertEquals(45, (int)props.get("prop2", 0))  
        );  
    }  
}
```

Assert multiple conditions together



#4 Precise exception handling

```
@Test
void testException() {
    MyClass underTest = new MyClass();
    assertEquals(42, underTest.theAnswer());

    assertThrows(FileNotFoundException.class, () -> {
        underTest.readFile();
    });
}
```

Instead of
@Test(expected= ...)

```
@Test
void testTimeout() {
    MyClass underTest = new MyClass();

    assertTimeout(Duration.ofMillis(500), () -> {
        underTest.longRunningOperation();
    });
}
```

Instead of
@Test(timeout= ...)

#5 Tagging

Allows to filter tests by tags

```
class TaggingTest {  
  
    @Test  
    @Tag("fast")  
    void testFastOperation() {  
        assertEquals(42, new MyClass().theAnswer());  
    }  
  
    @Test  
    @Tag("slow")  
    void testSlowOperation() {  
        new MyClass().longRunningOperation();  
    }  
  
}
```


#6 Dependency Injection

Inject dependencies from extensions or JUnit

```
@ExtendWith(AemContextExtension.class)
class DependencyInjectionTest {

    @Test
    void testDependencies(AemContext context, TestInfo testInfo) {
        MyClass underTest = new MyClass();

        context.registerService(MyClass.class, underTest);

        assertEquals(42, new MyClass().theAnswer(),
            "The answer was wrong in: " + testInfo.getDisplayName());
    }
}
```

#7 Conditional Test Execution

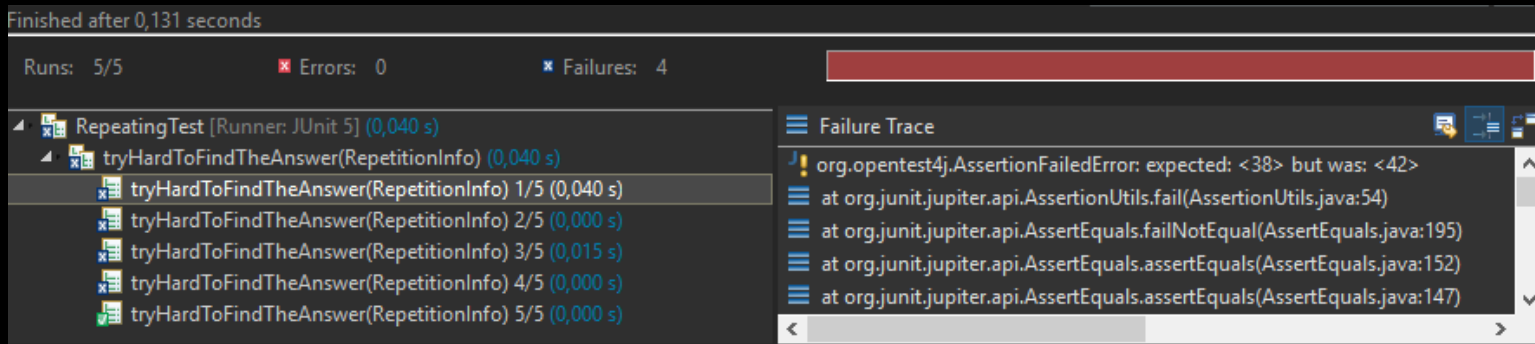
```
class ConditionalTest {  
  
    @Test  
    void runAlways() { /*...*/ }  
  
    @Test  
    @Disabled  
    void runNever() { /*...*/ }  
  
    @Test  
    @EnabledOnOs({ OS.LINUX, OS.MAC })  
    void runOnSpecificOS() { /*...*/ }  
  
    @Test  
    @DisabledOnJre(JRE.JAVA_10)  
    void runOnSpecificJRE() { /*...*/ }  
  
}
```

Environment-based
conditions
@Disabled... and
@Enabled...

#8 Repeated Test

Repeat 5 times

```
class RepeatingTest {  
  
    @RepeatedTest(value = 5, name = "{displayName} {currentRepetition}/{totalRepetitions}")  
    void tryHardToFindTheAnswer(RepetitionInfo info) {  
        assertEquals(37 + info.getCurrentRepetition(), new MyClass().theAnswer());  
    }  
  
}
```



Finished after 0,131 seconds

Runs: 5/5 ❌ Errors: 0 ❌ Failures: 4

RepeatingTest [Runner: JUnit 5] (0,040 s)

- tryHardToFindTheAnswer(RepetitionInfo) (0,040 s)
 - tryHardToFindTheAnswer(RepetitionInfo) 1/5 (0,040 s)
 - tryHardToFindTheAnswer(RepetitionInfo) 2/5 (0,000 s)
 - tryHardToFindTheAnswer(RepetitionInfo) 3/5 (0,015 s)
 - tryHardToFindTheAnswer(RepetitionInfo) 4/5 (0,000 s)
 - tryHardToFindTheAnswer(RepetitionInfo) 5/5 (0,000 s)

Failure Trace

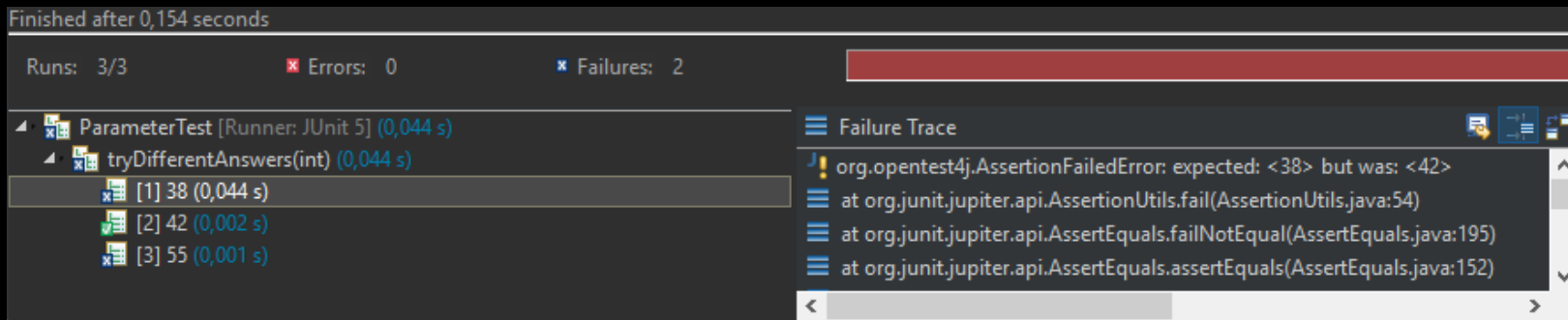
```
org.opentest4j.AssertionFailedError: expected: <38> but was: <42>  
    at org.junit.jupiter.api.AssertionsUtils.fail(AssertionsUtils.java:54)  
    at org.junit.jupiter.api.Assertions.failNotEqual(Assertions.java:195)  
    at org.junit.jupiter.api.Assertions.assertEquals(Assertions.java:152)  
    at org.junit.jupiter.api.Assertions.assertEquals(Assertions.java:147)
```

#9 Parameterized Tests

```
class ParameterTest {  
  
    @ParameterizedTest  
    @ValueSource(ints = { 38, 42, 55 })  
    void tryDifferentAnswers(int anAnswer) {  
        assertEquals(anAnswer, new MyClass().theAnswer());  
    }  
  
}
```

Run with parameters

- @ValueSource
- @EnumSource
- @MethodSource
- @CsvSource
- ...



Finished after 0,154 seconds

Runs: 3/3 ✖ Errors: 0 ✖ Failures: 2

ParameterTest [Runner: JUnit 5] (0,044 s)

- tryDifferentAnswers(int) (0,044 s)
 - [1] 38 (0,044 s)
 - [2] 42 (0,002 s)
 - [3] 55 (0,001 s)

Failure Trace

```
org.opentest4j.AssertionFailedError: expected: <38> but was: <42>  
at org.junit.jupiter.api.AssertionsUtils.fail(AssertionsUtils.java:54)  
at org.junit.jupiter.api.Assertions.failNotEqual(Assertions.java:195)  
at org.junit.jupiter.api.Assertions.assertEquals(Assertions.java:152)
```

#10 Meta annotations

Your own
“meta
annotation”

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@ExtendWith(AemContextExtension.class)
@ExtendWith(MockitoExtension.class)
public @interface AemContextAndMockito {}
```

Assigning it
applies all
extensions

```
@AemContextAndMockito
class MetaAnnotationTest {
    @Mock
    private MyService service;
    @Test
    void testRegisterMockService(AemContext context) {
        context.registerService(MyService.class, service);
        assertEquals(service, context.getService(MyService.class));
    }
}
```

#10 Meta annotations

Tagging via
meta
annotations

```
@Target({ ElementType.TYPE, ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
@Tag("fast")
public @interface Fast {}

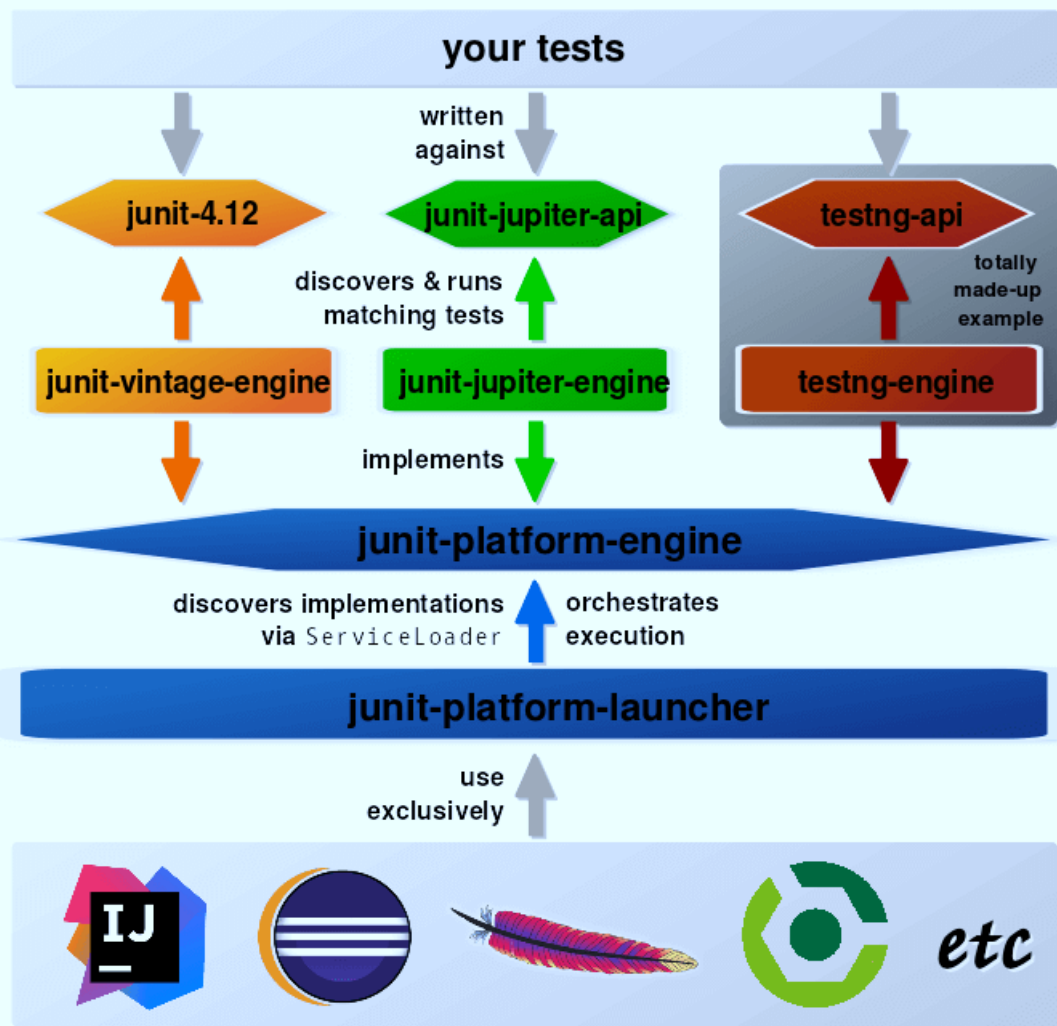
@Target({ ElementType.TYPE, ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
@Tag("slow")
public @interface Slow {}

class MetaAnnotationTagTest {
    @Test @Fast
    void testFastOperation() {
        assertEquals(42, new MyClass().theAnswer());
    }
    @Test @Slow
    void testSlowOperation() {
        new MyClass().longRunningOperation();
    }
}
```

JUnit 5 – Architecture

JUnit The JUnit logo consists of the word "JUnit" in a grey, sans-serif font, followed by a circular icon. The icon is split vertically: the left half is red and the right half is green. A white number "5" is centered within the circle.

Modular Architecture



- “Prefer extension points over features”
- Very good extensibility
- Example:
 - JUnit 5 provides only “classic” assertions
 - Use libs like AssertJ, Hamcrest, Thruth for complex assertions

JUnit 5 – Tooling

JUnit The JUnit logo consists of the word "JUnit" in a light gray, sans-serif font, followed by a circular icon. The icon is split vertically: the left half is red and the right half is green. A white number "5" is centered within the circle.

JUnit 5 Tooling support

- Maven Surefire Plugin ✓
- Mockito ✓
- IDEs like Eclipse, IntelliJ ✓
- Plugins for Eclipse Infinitest ✓ MoreUnit ✗
- Sling Mocks, AEM Mocks ✓

Maven Dependencies

“Bill of Material”
Defines all
versions

```
<dependencyManagement>
<dependencies>

<dependency>
<groupId>org.junit</groupId>
<artifactId>junit-bom</artifactId>
<version>5.3.0</version>
<type>pom</type>
<scope>import</scope>
</dependency>

</dependencies>
</dependencyManagement>
```

JUnit 4 and 5
side-by-side

```
<dependencies>
<dependency>
<groupId>org.junit.jupiter</groupId>
<artifactId>junit-jupiter-api</artifactId>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.junit.jupiter</groupId>
<artifactId>junit-jupiter-params</artifactId>
<scope>test</scope>
</dependency>
<dependency>
<groupId>org.junit.jupiter</groupId>
<artifactId>junit-jupiter-engine</artifactId>
<scope>test</scope>
</dependency>

<!-- Only required for JUnit 4 compatibility -->
<dependency>
<groupId>org.junit.vintage</groupId>
<artifactId>junit-vintage-engine</artifactId>
<scope>test</scope>
</dependency>
</dependencies>
```

JUnit 5 – Sling & AEM Mocks

JUnit The JUnit logo consists of the word "JUnit" in a grey, sans-serif font, followed by a circular icon. The icon is a circle divided vertically into two halves: the left half is red and the right half is green. A white number "5" is centered within the circle.

- JUnit 5 support since August 2018
- We got “modular”, too:
 - Core
 - JUnit 4
 - JUnit 5

```
<dependency>  
  <groupId>io.wcm</groupId>  
  <artifactId>io.wcm.testing.aem-mock.junit5</artifactId>  
  <version>2.3.2</version>  
</dependency>
```

```
<dependency>  
  <groupId>io.wcm</groupId>  
  <artifactId>io.wcm.testing.aem-mock.junit4</artifactId>  
  <version>2.3.2</version>  
</dependency>
```

Using AemContext in JUnit 5

Use "junit5" package

Define
@ExtendWith

```
import io.wcm.testing.mock.aem.junit5.AemContext;  
import io.wcm.testing.mock.aem.junit5.AemContextExtension;  
  
@ExtendWith(AemContextExtension.class)  
class MyUnitTest {
```

No @Rule
required

```
    private AemContext context = new AemContext();  
  
    @BeforeEach  
    void setUp() { /*...*/ }  
  
    @Test  
    void testMyCode() { /*...*/ }  
  
}
```

Optional:
RESOURCERESOLVER_MOCK
JCR_MOCK
JCR_OAK

Using AemContext as method parameter

```
import io.wcm.testing.mock.aem.junit5.AemContext;
import io.wcm.testing.mock.aem.junit5.AemContextExtension;

@ExtendWith(AemContextExtension.class)
class MyUnitTest {

    @BeforeEach
    void setUp(AemContext context) {
        // ...
    }

    @Test
    void testMyCode(AemContext context) {
        // ...
    }
}
```

Alternative:
Pass in AemContext
as method parameter

You can also use:
JcrMockAemContext
JcrOakAemContext
to switch resource
resolver type *

* Switching resource resolver type via method parameters currently only supported in AEM Mocks

Contributed by
Karol
Lewandowski

Using AemContextBuilder

```
@ExtendWith(AemContextExtension.class)
class MyUnitTest {

    private AemContext context = new AemContextBuilder()
        // register plugins
        .plugin(CACONFIG)
        .plugin(WCMIO_CACONFIG)
        // shared context setup code for all tests
        .<AemContext>afterSetUp(context -> {
            // register sling models
            context.addModelsForPackage("to.adapt.junit5demo");
        })
        .build();

    @BeforeEach
    void setUp() { /*...*/ }

    @Test
    void testMyCode() { /*...*/ }
}
```

Using
AemContextBuilder
not possible when
passing in as method
parameters

Recommendation:
make this code reusable
in a separate class

Migrating AEM/Sling Mocks Unit Tests

- It's easy to migrate JUnit 4 tests to JUnit 5
- See wiki page
<https://wcm-io.atlassian.net/wiki/x/AYAmJ>

- Migrate AEM Mock Unit Test to JUnit 5

A large, light blue circle with a white border, containing the text "LIVE DEMO" in white, uppercase, sans-serif font, centered within the circle.

LIVE
DEMO

Same for OSGi Mock, Sling Mock

- In slides an demo we used only AEM Mocks
- But same applies also when using
 - OSGi Mocks
 - Sling Mocks

- What JUnit 5 features do **you** want to use with AEM/Sling Mocks?
- Get in touch with us (mailing lists)

References

JUnit 5

Related adaptTo() talks

- adaptTo() 2016:
[Unit Testing with Sling & AEM Mocks](#)
- adaptTo() 2015:
[So how do I test my Sling application?](#)
- adaptTo() 2014:
[Lightning Talk: Mock AEM & Co for Unit Tests](#)

- **wcm.io AEM Mocks**
<http://wcm.io/testing/aem-mock/>
- **Apache Sling Mocks**
<http://sling.apache.org/documentation/development/sling-mock.html>
<http://sling.apache.org/documentation/development/osgi-mock.html>
<http://sling.apache.org/documentation/development/jcr-mock.html>
- **JUnit 5**
<https://junit.org/junit5/>
- **Need Help? Use the Apache Sling or wcm.io Mailing Lists**
<https://sling.apache.org/project-information.html#mailing-lists>
<http://wcm.io/mailling-lists.html>
- **Demo code for this talk**
<https://github.com/adaptto/2018-junit5-and-sling-aem-mocks>