JUnit 6.0

Marc Philipp

Software Engineer (self-employed)

JUnit committer since 2012
team lead since 2016

Mastodon: @marcphilipp@chaos.social
Bluesky: @marcphilipp.de
Web: marcphilipp.de
Email: marc@junit.org

JUnit in a nutshell

- originally created by Kent Beck and Erich Gamma in 1997
- *independent* open source project
- not backed by a company or part of a foundation
- most work happens in the spare time of contributors
- one of the most popular Java libraries

Thank you to our sponsors!

https://junit.org/sponsoring

The Sovereign Tech Fund is investing in JUnit in 2024 and 2025!

Sovereign Tech Fund

- German government program to strengthen the open source ecosystem
- Invests in foundational technologies that enable the creation of other software

https://www.sovereign.tech/programs/fund

JUnit Generations

JUnit Generations

4.x

Java 5 baseline – maintenance mode

JUnit Generations

4.x

Java 5 baseline – maintenance mode

5.x

Java 8 baseline – initial release in 2017

JUnit Generations

4.x

Java 5 baseline – maintenance mode

5.x

Java 8 baseline – initial release in 2017

6.x

Java 17 baseline – released on Sep 30, 2025

Full-time work on JUnit (2024-2025)

Full-time work on JUnit (2024-2025)

- Better test reports ✅ 5.12

Full-time work on JUnit (2024-2025)

- Better test reports ✅ 5.12
- Parameterized test classes ✅ 5.13

Full-time work on JUnit (2024-2025)

- Better test reports ✅ 5.12
- Parameterized test classes ✅ 5.13
- Reporting of test discovery issues ✅ 5.13

Full-time work on JUnit (2024-2025)

- Better test reports ✅ 5.12
- Parameterized test classes ✅ 5.13
- Reporting of test discovery issues ✅ 5.13
- Better Kotlin support ✅ 6.0

Full-time work on JUnit (2024-2025)

- Better test reports ✅ 5.12
- Parameterized test classes ✅ 5.13
- Reporting of test discovery issues ✅ 5.13
- Better Kotlin support ✅ 6.0
- Safe cancellation of test execution ✅ 6.0

Full-time work on JUnit (2024-2025)

- Better test reports ✅ 5.12
- Parameterized test classes ✅ 5.13
- Reporting of test discovery issues ✅ 5.13
- Better Kotlin support ✅ 6.0
- Safe cancellation of test execution ✅ 6.0
- Improved parallel test execution ✅ 6.1

Full-time work on JUnit (2024-2025)

- Better test reports ✅ 5.12
- Parameterized test classes ✅ 5.13
- Reporting of test discovery issues ✅ 5.13
- Better Kotlin support ✅ 6.0
- Safe cancellation of test execution ✅ 6.0
- Improved parallel test execution ✅ 6.1
- Improved documentation ⚙️ 6.1

Today's Agenda

1. Better test reports
2. Parameterized test classes
3. Reporting of test discovery issues
4. JUnit 6.0

More info

- Blog post series: https://urlr.me/AwhBSm
- JetBrains IDEA Conf 2025 talk: https://youtu.be/-cY5BJaHz6E

Better Test Reports

Open Test Reporting XML format 5.9

- Replacement for legacy XML format that lacks support for many JUnit Platform features
- Requires runtime dependency on `junit-platform-reporting`
- Enabled via configuration parameter `junit.platform.reporting.open.xml.enabled=true`

Event-based format 5.9

```
<?xml version="1.0" ?>
<e:events xmlns="https://schemas.opentest4j.org/reporting/core/0.1.0" xmlns:e="https://schemas.oper
<infrastructure><hostName>...</hostName><userName>marc</userName><operatingSystem>Mac OS X</operati
<e:started id="766" name="JUnit Jupiter" time="2022-09-23T07:54:50.324086Z"><metadata><junit:unique
<e:started id="767" name="ColorPaletteTests" parentId="766" time="2022-09-23T07:54:50.324275Z"><met
<e:started id="768" name="DemonstratePalettesTests" parentId="767" time="2022-09-23T07:54:50.32445
<e:started id="769" name="flat_default()" parentId="768" time="2022-09-23T07:54:50.324589Z"><metada
<e:finished id="769" time="2022-09-23T07:54:50.326039Z"><result status="SUCCESSFUL"></result></e:fi
<e:finished id="768" time="2022-09-23T07:54:50.332254Z"><result status="SUCCESSFUL"></result></e:fi
<e:finished id="767" time="2022-09-23T07:54:50.333862Z"><result status="SUCCESSFUL"></result></e:fi
<e:finished id="766" time="2022-09-23T07:54:50.812066Z"><result status="SUCCESSFUL"></result></e:fi
</e:events>
```

👍 Suitable for writing and streaming

👎 Not very human-readable

Hierarchical format 5.9

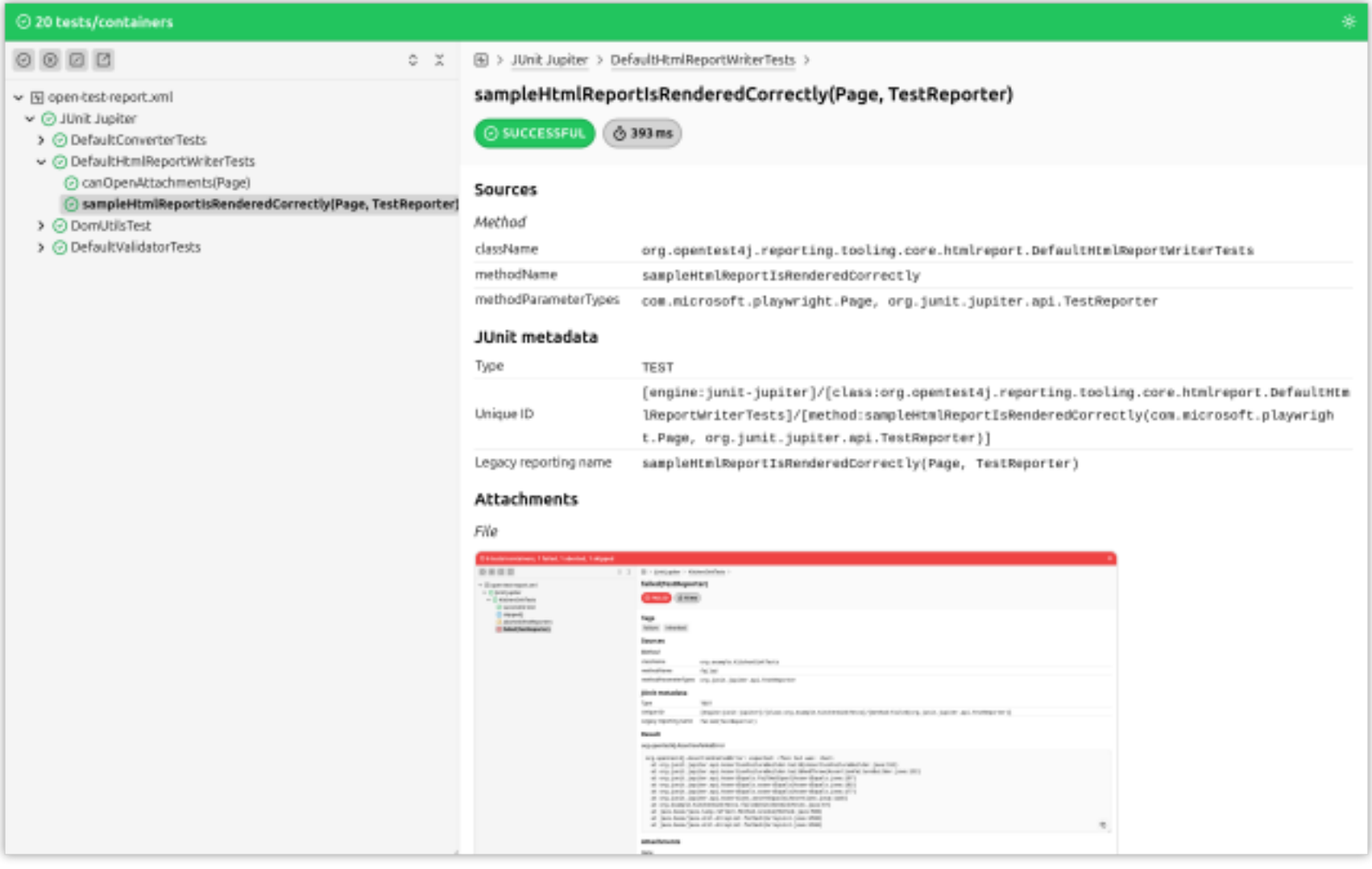
Converted from event-based format via CLI tool

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<h:execution xmlns:h="https://schemas.opentest4j.org/reporting/hierarchy/0.1.0" xmlns="https://sche
  <infrastructure><!-- ... --></infrastructure>
  <h:root duration="PT0.48798S" name="JUnit Jupiter" start="2022-09-23T07:54:50.324086Z">
    <metadata>
      <junit:uniqueId>[engine:junit-jupiter]</junit:uniqueId>
      <junit:legacyReportingName>JUnit Jupiter</junit:legacyReportingName>
      <junit:type>CONTAINER</junit:type>
    </metadata>
    <result status="SUCCESSFUL"/>
    <h:child duration="PT0.009587S" name="ColorPaletteTests" start="2022-09-23T07:54:50.324275Z">
      <metadata><!-- ... --></metadata>
      <sources><java:classSource className="org.junit.platform.console.tasks.ColorPaletteTests"/></
      <result status="SUCCESSFUL"/>
      <h:child duration="PT0.007798S" name="DemonstratePalettesTests" start="2022-09-23T07:54:50.32
        <metadata><!-- ... --></metadata>
        <sources><java:classSource className="org.junit.platform.console.tasks.ColorPaletteTests$De
        <result status="SUCCESSFUL"/>
        <h:child duration="PT0.00145S" name="flat_default()" start="2022-09-23T07:54:50.324589Z">
          <metadata><!-- ... --></metadata>
          <sources><java:methodSource className="org.junit.platform.console.tasks.ColorPaletteTests
          <result status="SUCCESSFUL"/>
        </h:child>
      </h:child>
```

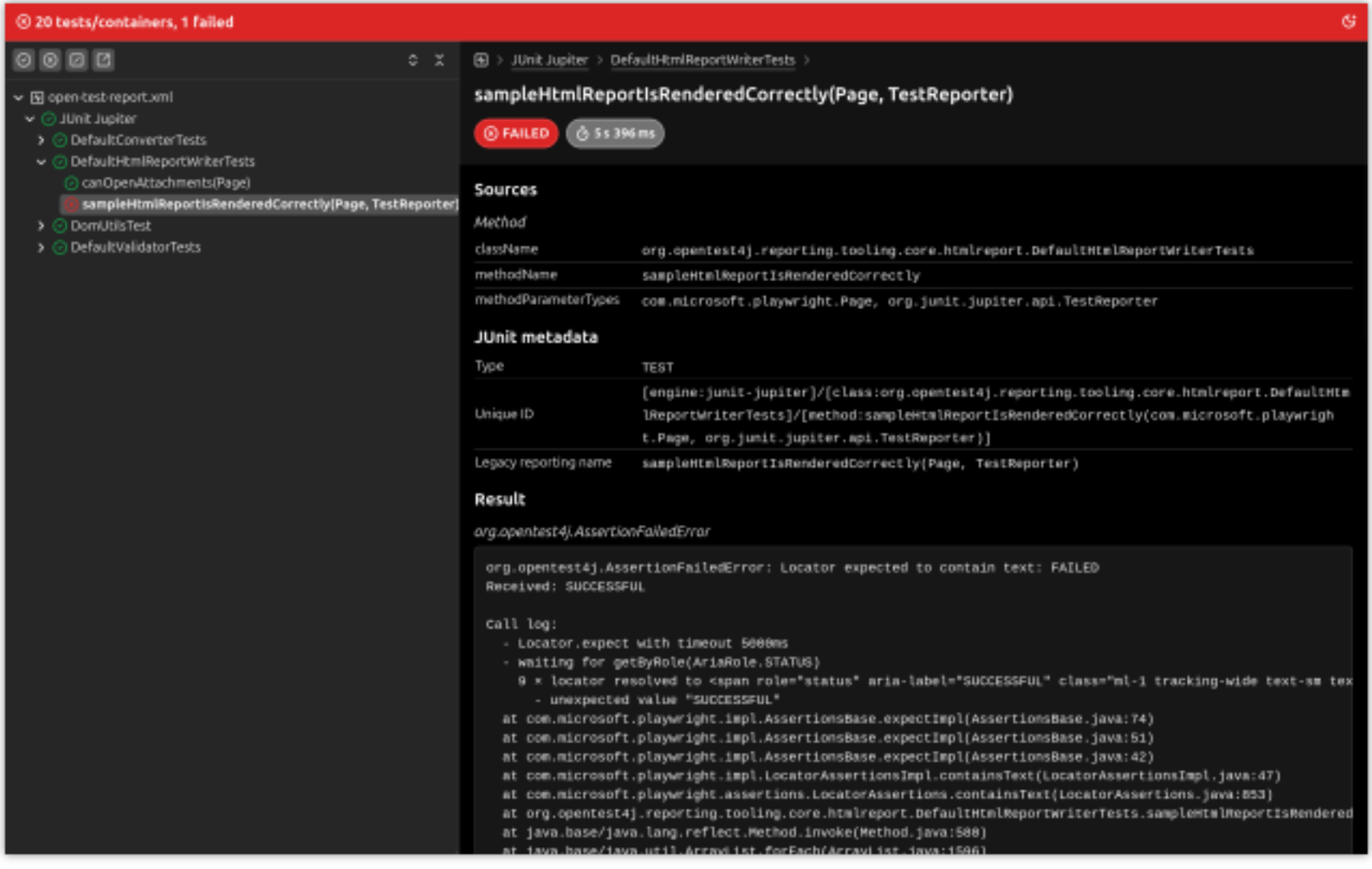
HTML Report 5.12

- XML is not very human-readable
- Introduce HTML report!

HTML Report 5.12



HTML Report 5.12



Demo

Part 2

Platform-agnostic and extensible

- Maintained in separate open-test-reporting repository
- Adoption in other ecosystems (e.g. PHPUnit) has started

Future work

- Gradle plugin?
- Maven plugin?
- GitHub Action?

Feedback?

👉 *open-test-reporting on GitHub*

Parameterized Test Classes

Parameterized Test Methods 5.0

```
class SomeTests {
    @ParameterizedTest
    @ValueSource(strings = {"foo", "bar"})
    void shouldNotBeNull(String value) {
        assertNotNull(value);
    }
    @ParameterizedTest
    @ValueSource(strings = {"foo", "bar"})
    void lengthShouldBeThree() {
        assertEquals(3, value.length());
    }
}
```

- Annotate *method* with `@ParameterizedTest`
- Add at least one `@...Source` annotation:

Parameterized Test Classes 5.13

```java
@ParameterizedClass
@ValueSource(strings = {"foo", "bar"})
class SomeTests {
    @Parameter String value;

    @Test
    void shouldNotBeNull() {
        assertNotNull(value);
    }
    @Test
    void lengthShouldBeThree() {
        assertEquals(3, value.length());
    }
}
```

- Annotate *class* with `@ParameterizedClass`
- Add at least one `@...Source` annotation

Consuming Arguments

- *Either* constructor *or* field injection
- Optional Java record support

```
@ParameterizedClass
@ValueSource(strings = {"foo", "bar"})
record SomeTests(String value) {
    @Test
    void shouldNotBeNull() {
        assertNotNull(value);
    }
    @Test
    void lengthShouldBeThree() {
        assertEquals(3, value.length());
    }
}
```

Demo

Mutable vs. immutable arguments

Prefer immutable data for `@ParameterizedClass` *arguments!*

- If they are *mutable*, tests may accidentally see modifications made by other tests
- You can use lifecycle methods (for example, `@AfterEach`) to reset data
  - Parallel execution would not be safe, though

Argument Source Annotations

- `@ValueSource`, `@EnumSource`, `@NullSource` 5.4, and `@EmptySource` 5.4 for simple values
- `@CsvSource`, `@CsvFileSource` for CSV content
- `@MethodSource`, `@FieldSource` 5.11 for custom code
- `@ArgumentsSource(MyProvider.class)` for custom providers

Lifecycle Methods

- Parameterized classes may declare `@BeforeParameterizedClassInvocation` and `@AfterParameterizedClassInvocation` lifecycle methods
- Called once before/after each invocation of the parameterized class with a set of arguments

```
@BeforeParameterizedClassInvocation
static void before(List<String> list) {
    if (list instanceof ArrayList<?> arrayList) {
        arrayList.ensureCapacity(100);
    }
}
```

Discovery Issues

Discovery issues?

Have you ever written a test and noticed at some later point that it wasn't actually being executed?

😱😱😱

Valid test methods?

```java
@Test // Java
int test() {
    return 42;
}
```

```kotlin
@Test // Kotlin
fun test(): Nothing = fail()
```

Valid test methods?

```java
@Test // Java
int test() {
    return 42;
}
```

```kotlin
@Test // Kotlin
fun test(): Nothing = fail()
```

`@Test` methods must be declared having a `void` (`Unit` in Kotlin) return type!

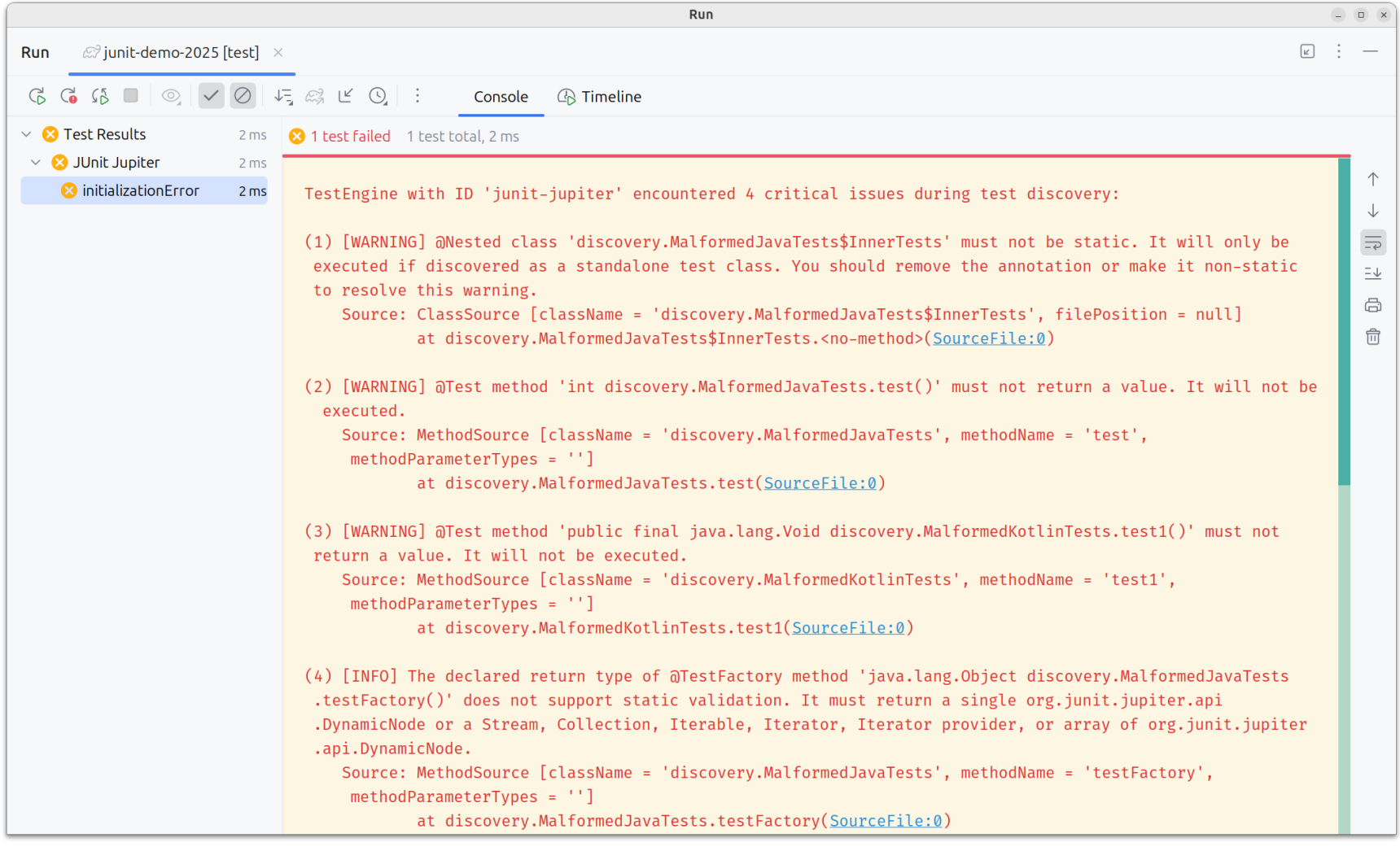
Malformed declarations

- Test classes and methods are checked during test discovery phase
- Before 5.13, any malformed declaration was silently discarded
- IDEs such as IntelliJ often provide inspections to help avoid running into such issues

Discovery issue reporting 5.13

- Test engines can report issues such as malformed declarations during test discovery
- Non-critical issues are logged
- Critical issues cause test execution to fail for that test engine
- Critical severity is configurable via configuration parameter `junit.platform.discovery.issue.severity.critical`

Discovery issue reporting 5.13



Demo

JUnit 6.0

Overview of 6.0

- No general API changes
  - JUnit Jupiter API for writing tests and extensions
  - JUnit Platform APIs for IDE and build tool integration
- Migration from 5.x to 6.0 will be much easier than from 4.x to 5.0

Major changes in 6.0

Major changes in 6.0

- Require Java 17+

Major changes in 6.0

- Require Java 17+
- Require Kotlin 2.1+

Major changes in 6.0

- Require Java 17+
- Require Kotlin 2.1+
- Add `@Nullable` information via JSpecify annotations

Major changes in 6.0

- Require Java 17+
- Require Kotlin 2.1+
- Add `@Nullable` information via JSpecify annotations
- Custom JFR events built-in

Major changes in 6.0

- Require Java 17+
- Require Kotlin 2.1+
- Add `@Nullable` information via JSpecify annotations
- Custom JFR events built-in
- Remove APIs deprecated in 5.10 and earlier

Major changes in 6.0

- Require Java 17+
- Require Kotlin 2.1+
- Add `@Nullable` information via JSpecify annotations
- Custom JFR events built-in
- Remove APIs deprecated in 5.10 and earlier
  - Discontinue junit-platform-runner

Major changes in 6.0

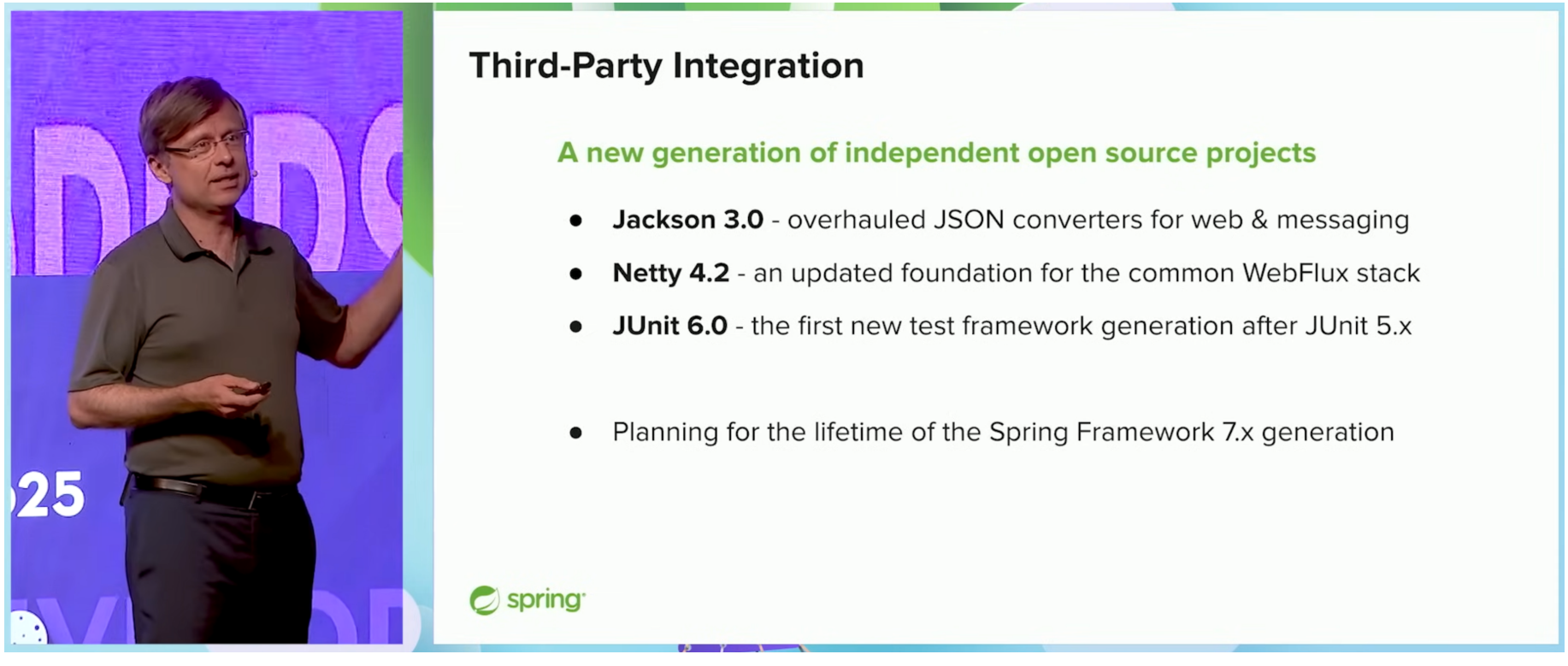
- Require Java 17+
- Require Kotlin 2.1+
- Add `@Nullable` information via JSpecify annotations
- Custom JFR events built-in
- Remove APIs deprecated in 5.10 and earlier
  - Discontinue junit-platform-runner
- Deprecate junit-vintage-engine and junit-jupiter-migrationsupport

Long(er)-Term Support for 5.14.x

Don’t panic!

- Stays on Java 8 baseline
- Critical bugs will be fixed

Spring Framework 7.0 and Spring Boot 4.0 are based on JUnit 6.0



Announced by Jürgen Höller at Spring I/O

Example Code and Slides

- Slides: https://marcphilipp.de/en/talks/
- Code: https://github.com/marcphilipp/junit-demo-2025

Q & A

More questions?

✉ marc@junit.org

Thanks!