JUnit 5 Release Notes

Stefan Bechtold, Sam Brannen, Johannes Link, Matthias Merdes, Marc Philipp, Christian Stein

Version 5.5.0-SNAPSHOT

Table of Contents

5.4.0 1
- JUnit Platform 1
- JUnit Jupiter 3
- JUnit Vintage 5

5.3.2 5
- JUnit Platform 5
- JUnit Jupiter 5
- JUnit Vintage 6

5.3.1 6
- JUnit Platform 6
- JUnit Jupiter 6
- JUnit Vintage 7

5.3.0 7

This document contains the *change log* for all JUnit 5 releases since 5.3 GA.

Please refer to the User Guide for comprehensive reference documentation for programmers writing tests, extension authors, and engine authors as well as build tool and IDE vendors.

5.4.0

Date of Release: February 7, 2019

Scope:

- New `junit-jupiter` dependency-aggregating artifact for simplified dependency management in build tools
- XML report generating listener
- Test Kit for testing engines and extensions
- `null` and *empty* argument sources for `@ParameterizedTest` methods
- `@TempDir` support for temporary directories
- Custom display name generator API
- Support for ordering test methods
- Support for ordering extensions registered via `@RegisterExtension`
- `TestWatcher` extension API
- API for accessing outer test instances in `ExtensionContext`
- JUnit 4 `@Ignore` migration support
- Improved diagnostics and error reporting
- Improved documentation and user experience in the User Guide
- Discontinuation of the `junit-platform-surefire-provider`
- Various minor improvements and bug fixes

For a complete list of all *closed* issues and pull requests for this release, consult the 5.4 M1, 5.4 RC1, 5.4 RC2, and 5.4 GA milestone pages in the JUnit repository on GitHub.

JUnit Platform

Bug Fixes

- The `junit-platform-suite-api` artifact no longer has an unnecessary direct dependency on `junit-platform-commons`.
- Containers and tests that interrupt the current thread no longer cause surprising failures in subsequent tests that interact with the reused thread.
    - This applies specifically to containers and tests executed via a `HierarchicalTestExecutorService` — for example, a `@Test` method in JUnit Jupiter — that

interrupt the current thread — for example, via `Thread.currentThread().interrupt()` — but fail to clear the *interrupted status* flag for the current thread.

  - The `TestTask` implementation used internally by `HierarchicalTestExecutorService` implementations now automatically clears the *interrupted status* for the current thread after the execution of each container and test and logs a message at `DEBUG` level (`FINE` in java.util.logging) if the *interrupted status* is not cleared properly by user code.

Deprecations and Breaking Changes

- The JUnit Platform Maven Surefire Provider (`junit-platform-surefire-provider`) has been discontinued and is no longer released as part of JUnit 5. Please use Maven Surefire's native support for running tests on the JUnit Platform instead (requires Maven Surefire 2.22.0 or higher).
- `TestPlan.add(TestIdentifier)` has always been considered *internal* and is now *deprecated*. While calls from third-party code will continue to work for the time being, IDEs and build tools should remove any such code as soon as possible.

New Features and Improvements

- New `junit-platform-reporting` artifact containing a `LegacyXmlReportGeneratingListener` that generates XML reports using a format which is compatible with the de facto standard for JUnit 4 based test reports that was made popular by the Ant build system.
  - See JUnit Platform Reporting in the User Guide for details.
- New `junit-platform-testkit` artifact containing a *Test Kit* API for testing the execution of a `TestEngine` running on the JUnit Platform.
  - See JUnit Platform Test Kit in the User Guide for details.
- If a container or test fails, the root cause and suppressed exceptions are now included in the stack trace printed by the `ConsoleLauncher`.
- The `ConsoleLauncher` now sanitizes user-provided display names before printing them to the console.
  - Common whitespace characters such as `\t`, `\n`, `\x0B`, `\f`, and `\r` are replaced by a standard space character, and all other ISO control characters are emitted as a single dot (`.`).
- New `ModifierSupport` class providing an API for extension and test engine authors to inspect modifiers of classes and members.
  - See the User Guide for details.
- `AnnotationSupport` provides new methods for finding annotated fields and their values.
  - Consult the Javadoc for the various `findAnnotatedFields()` and `findAnnotatedFieldValues()` methods in `AnnotationSupport` for details.
- `ReflectionSupport` provides new methods for finding fields and reading a field's value.
  - Consult the Javadoc for the `findFields()` and `tryToReadFieldValue()` methods in `ReflectionSupport` for details.
- Exceptions reported due to failed reflective operations such as loading a class, reading a field's

value, or looking up a method by name now include the original exception as their cause to make it easier to debug underlying issues.

- The `Launcher` API now provides an `execute(TestPlan, TestExecutionListener…)` method that allows one to execute a previously discovered `TestPlan`.
- `@RunWith(JUnitPlatform.class)` no longer executes test discovery twice.
- Implementations of `HierarchicalTestEngine` may now add behavior that wraps around the invocation of `Node.before()`, `Node.execute()`, and `Node.after()` using the new `Node.around()` hook.

JUnit Jupiter

Bug Fixes

- `@RegisterExtension` fields that are `null` when evaluated are no longer silently ignored. Instead, the corresponding test class or test method now fails with an informative exception.
- `@ParameterizedTest` once again supports `MessageFormat` patterns for individual parameters — for example, `{0,number,#.##}`.
- `@ResourceLock` can now be declared on test template methods such as `@RepeatedTest` and `@ParameterizedTest` methods. If `@ResourceLock` is used, the invocations will run in the same thread, even if parallel execution is enabled.
- The `@Execution` and `@ResourceLock` annotations used to configure parallel test execution are now inherited within test class hierarchies (via `@Inherited` semantics).

Deprecations and Breaking Changes

- The default mode for parallel test execution has been changed from `CONCURRENT` to `SAME_THREAD` to allow for gradual opt-in by using the `@Execution` annotation on individual test classes or methods. You can invert this behavior by changing the default execution mode via the new `junit.jupiter.execution.parallel.mode.default` configuration parameter. Please refer to the User Guide for details.

New Features and Improvements

- New `org.junit.jupiter:junit-jupiter` artifact that simplifies dependency management for JUnit Jupiter in build tools such as Gradle and Maven.
  - Specifically, this artifact aggregates all dependencies that are required to use JUnit Jupiter along with optional dependencies that extend the core Jupiter APIs.
  - It contains compile-time dependencies on `junit-jupiter-api` and `junit-jupiter-params` and a runtime dependency on `junit-jupiter-engine`.
- `Assertions.assertEquals()` variants that compare floating point numbers using a delta now support a *delta* of zero.
- New `Assertions.assertEquals()` variants that accept mixed boxed and unboxed primitive values, allowing statements such as `assertEquals(42, Integer.valueOf("42"))` to compile.
- New `Assertions.assertNotEquals()` variants that accept the following primitive data types: `char`,

`byte`, `short`, `int`, `long`, `float`, and `double`. Mixed boxed and unboxed primitive values are also supported.

- Exceptions thrown in `Assertions.assertAll()` are now additionally tracked as *suppressed exceptions* in the resulting `MultipleFailuresError`. Consequently, the stack traces for such exceptions are now visible as *Suppressed* at the end of the stack trace for the invocation of `assertAll()`.
- JUnit 4's `AssumptionViolatedException` is now supported in JUnit Jupiter for aborting a test mid-flight due to a failed assumption — for example, via JUnit 4's `org.junit.Assume` utility class.
- JUnit 4's `@Ignore` annotation is now supported for disabling test classes and test methods via the `junit-jupiter-migrationsupport` module.
  - See the User Guide for details.
- New `@TempDir` extension (formerly part of JUnit Pioneer) that allows one to write tests that require a temporary directory in a `java.nio.file.FileSystem`.
  - See the User Guide for details.
- In addition to returning streams, `@TestFactory`-annotated methods may now return a single `DynamicNode` — for example, a `DynamicTest` or a `DynamicContainer`.
- New `@NullSource`, `@EmptySource`, and `@NullAndEmptySource` argument sources that provide `null` and *empty* arguments to `@ParameterizedTest` methods.
  - See Null and Empty Sources in the User Guide for details.
- Implicit conversion from hexadecimal and octal string representations to integral types in `@ParameterizedTest` arguments — for example, conversion from `"0xff"` to `255`.
- New `JAVA_12` and `JAVA_13` constants in the `JRE` enum for use with `@EnabledOnJre` and `@DisabledOnJre`.
- New `LOCALE` and `TIME_ZONE` constants in `org.junit.jupiter.api.parallel.Resources` for use with `@ResourceLock` to synchronize test execution regarding the default `Locale` and default `TimeZone`, respectively.
- New `MethodOrderer` API for ordering the sequence of tests with built-in support for *alphanumeric*, `@Order` annotation based, and *random* ordering of test methods.
  - See Test Execution Order in the User Guide for details.
- New `DisplayNameGenerator` interface and `@DisplayNameGeneration` annotation that allow declarative configuration of a pre-defined or custom display name generator.
  - See Display Name Generators in the User Guide for details.
- New `TestWatcher` extension API that allows extensions to process test results by defining result-based callbacks invoked after text execution.
  - See Test Result Processing in the User Guide for details.
- Extensions registered *programmatically* via `@RegisterExtension` may now be registered in an explicit order via the `@Order` annotation.
  - See Extension Registration Order in the User Guide for details.
- New `ExtensionContext` methods to access all test instances, including enclosing ones for `@Nested` tests: `getTestInstances()` and `getRequiredTestInstances()`.

JUnit Vintage

Bug Fixes

- The `VintageTestEngine` now uses the fully qualified class name as the *legacy reporting name* for Vintage test classes instead of the simple class name which caused problems in test reports based on legacy reporting names — for example, reports generated by Maven Surefire.

New Features and Improvements

- The `VintageTestEngine` now validates that the version of `junit:junit` on the classpath is supported (i.e., is equal to or greater than 4.12).

5.3.2

Date of Release: November 25, 2018

Scope: Bug fixes since 5.3.1

For a complete list of all *closed* issues and pull requests for this release, consult the 5.3.2 milestone page in the JUnit repository on GitHub.

JUnit Platform

Bug Fixes

- When configured with `--details verbose`, the `ConsoleLauncher` no longer throws a `MissingFormatArgumentException` if a test method display name contains `String` format specifiers such as `%c`.

JUnit Jupiter

Bug Fixes

- `Assertions.assertAll()` is now thread-safe — for example, it can now be used with a *parallel* `Stream`.
- The `OS.SOLARIS` enum constant used with `@EnabledOnOs` and `@DisabledOnOs` is now also detected as the current operating system if the `os.name` JVM system property contains `"SunOs"`.
- `Assertions.assertLinesMatch()` no longer throws a `NullPointerException` after evaluating a fast-forward match if there are more expected lines after the fast-forward match than remain in the actual results. This bug only manifested itself if the expected list size was equal to or greater than the actual list size.
- Multidimensional arrays may now be supplied to `@ParameterizedTest` methods from factory methods configured via `@MethodSource`.
    - For example, a factory method with the signature `static Stream<int[][]> factory()` can be

used as the `@MethodSource` for a `@ParameterizedTest` with the signature `void test(int[][])`.

- Threads created for running tests in parallel now use the same thread context class loader (*TCCL*) that was set when creating the underlying executor service. This resolves `ClassNotFoundException` issues that only occur in parallel execution mode when a custom *TCCL* is in place.
- When executing tests in parallel, lifecycle methods and callbacks called after a `@TestFactory` method are now always executed after the dynamic tests returned by the method.
- Exceptions thrown during initialization of static `@RegisterExtension` fields now cause the test class to fail instead of being silently swallowed.

JUnit Vintage

No changes

5.3.1

Date of Release: September 11, 2018

Scope: Bug fixes since 5.3.0

For a complete list of all *closed* issues and pull requests for this release, consult the 5.3.1 milestone page in the JUnit repository on GitHub.

JUnit Platform

Bug Fixes

- An `OutOfMemoryError` regression introduced in JUnit 5.3.0 has been fixed.
  - Specifically, the `NodeTestTask` used by implementations of `HierarchicalTestEngine` (such as the Jupiter and Vintage test engines) no longer retains references to contextual state after a node has completed execution. This allows state such as instances of test classes to be properly garbage collected by the JVM.
  - Previously, a `NodeTestTask` instance was created for each `TestDescriptor` before starting execution. Now they are created on the fly and can be garbage collected by the JVM after the enclosing container has finished.
- The OpenTest4J dependency has been updated to 1.1.1 to fix a serialization incompatibility between 1.0.0 and 1.1.0 that caused failure messages to be discarded when used from Gradle and potentially other tools and IDEs.

JUnit Jupiter

Bug Fixes

- Invocations of `assertThrows()` that are passed a method reference for an overloaded method

with a `void` return type once again compile.

  - For example, given an instance of `java.lang.Object` stored in a variable named `object`, `assertThrows(Exception.class, object::wait)` compiled against JUnit 5.2.0, failed to compile against JUnit 5.3.0, but now compiles against JUnit 5.3.1.

Breaking Changes

- In order to revert the aforementioned breaking change, variants of `assertThrows()` introduced in JUnit 5.3.0 that accept `ThrowingSupplier` arguments have been removed.

JUnit Vintage

No changes

5.3.0

Date of Release: September 3, 2018

Scope: Parallel test execution, output capture for `System.out` and `System.err`, new `TestInstanceFactory` extension API, custom test sources for dynamic tests, promotion of the dynamic test API from *experimental* to *maintained* status, discontinuation of the `junit-platform-gradle-plugin`, deprecation of the `junit-platform-surefire-provider`, as well as various minor improvements and bug fixes.

For complete details consult the 5.3.0 Release Notes online.