JUnit 5 Release Notes

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

Version 5.3.0-SNAPSHOT

Table of Contents

This document contains the *change log* for all JUnit 5 releases since 5.2 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.

5.3.0-RC1

Date of Release:

Scope:

For a complete list of all *closed* issues and pull requests for this release, consult the 5.3 RC1 milestone page in the JUnit repository on GitHub.

JUnit Platform

Bug Fixes

- `ClassSource` has been revised so that `equals()` and `hashCode()` are now properly based on the required *class name* instead of the optional `Class` reference. In addition, the non-blank precondition for a class name is now enforced.

Deprecations and Breaking Changes

-

New Features and Improvements

- New `MethodSource.from(Class, Method)` static factory method for creating a `MethodSource` from a specific class and method. This method should be used in favor of `MethodSource.from(Method)` when the test method is inherited from a superclass or present as an interface `default` method.
- A `ClasspathResourceSource` can now be created from a `URI` via the new `from(URI)` static factory method if the `URI` uses the `classpath` scheme.

JUnit Jupiter

Bug Fixes

- The `MethodSource` for an *inherited* `@Test` method now correctly references the *current* test class instead of the class or interface in which the `@Test` method is *declared*. This allows build tools such as Maven Surefire to properly include inherited `@Test` methods when executing a single test class or specific test classes based on filters — for example, via `mvn test -Dtest=SubclassTests`).
- Exceptions thrown in `@After` and `@AfterAll` lifecycle methods now take precedence over violated assumptions (i.e. `TestAbortedExceptions`) in test or prior lifecycle methods.

Deprecations and Breaking Changes

-

New Features and Improvements

- Generation of a detailed failure message for a failed assertion no longer fails if the `toString()` implementation of an object supplied to the assertion throws an exception. Instead, the object with the broken `toString()` implementation will be referenced via a default String representation based on the object's fully qualified class name and system hash code, separated by an `@` symbol.
- Although it is *highly discouraged*, it is now possible to extend the `org.junit.jupiter.api.Assertions` and `org.junit.jupiter.api.Assumptions` classes for special use cases.
- A custom test source `URI` for a dynamic container or dynamic test will now be registered as a `ClasspathResourceSource` if the `URI` uses the `classpath` scheme.
- New `TestInstanceFactory` extension API that enables custom creation of test class instances.
  - See Test Instance Factories in the User Guide for details.
- `ArgumentConverters` and `ArgumentsAggregators` registered using `@ConvertWith` and `@AggregateWith`, respectively, are now only instantiated once per `@ParameterizedTest` instead of once for each invocation.
- Performance improvements for executing parameterized tests, particularly when the method declares more than a few parameters.
- Generation of the display name for a `@ParameterizedTest` no longer fails if the `toString()` implementation of an argument for the parameterized test throws an exception. Instead, the object with the broken `toString()` implementation will be referenced via a default String representation based on the object's fully qualified class name and system hash code, separated by an `@` symbol.
- New `getAs<Class>(index)` Kotlin extension method to make `ArgumentsAccessor` friendlier to use from Kotlin.

JUnit Vintage

Bug Fixes

- The `MethodSource` for an *inherited* `@Test` method now correctly references the *current* test class instead of the class or interface in which the `@Test` method is *declared*. This allows build tools such as Maven Surefire to properly include inherited `@Test` methods when executing a single test class or specific test classes based on filters — for example, via `mvn test -Dtest=SubclassTests`).

Deprecations and Breaking Changes

-

New Features and Improvements

- The `VintageTestEngine` now uses the *simple name* of a test class as the display name instead of the *fully qualified class name*. This aligns with the behavior of the `JupiterTestEngine`.

5.3.0-M1

Date of Release: June 24, 2018

Scope: Parallel test execution, output capturing, test sources for dynamic tests as well as various minor improvements and bug fixes.

For a complete list of all *closed* issues and pull requests for this release, consult the 5.3 M1 milestone page in the JUnit repository on GitHub.

JUnit Platform

Bug Fixes

- The full stacktrace is now printed to the console when running the `ConsoleLauncher` in `--details verbose` mode.
- `ReflectionUtils.findNestedClasses()` and `ReflectionSupport.findNestedClasses()` no longer allow a `NoClassDefFoundError` to propagate if a nested class or nested interface has an invalid class file. Instead, the error will now be swallowed and logged at `WARNING` level.

Deprecations and Breaking Changes

- The `junit-platform-gradle-plugin` has been discontinued and is no longer released as part of JUnit 5. Please use Gradle's native support for running tests on the JUnit Platform (requires Gradle 4.6 or higher) instead.
- The `findAnnotation()` methods in `AnnotationSupport` and `AnnotationUtils` no longer cache annotation lookups. Note, however, that the algorithm remains otherwise unmodified and is therefore semantically identical to the previous behavior.

New Features and Improvements

- Experimental support for capturing output printed to `System.out` and `System.err` during test execution. This feature is disabled by default and can be enabled using a configuration parameter (see the User Guide for details).
- Reusable support for parallel test execution for test engines that extend `HierarchicalTestEngine`.
  - `HierarchicalTestEngine` implementations may now specify a `HierarchicalTestExecutorService`.
  - By default, a `SameThreadHierarchicalTestExecutorService` is used.
  - Test engines may use `ForkJoinPoolHierarchicalTestExecutorService` to support parallel test execution based on Java's Fork/Join framework.

    - `Node` implementations may provide a set of `ExclusiveResources` and an `ExecutionMode` to be used by `ForkJoinPoolHierarchicalTestExecutorService`.
- New `UriSource.from(URI)` static factory method that allows a `TestSource` to be created from a `URI`. If the `URI` references a file or directory in the local filesystem, a `FileSource` or `DirectorySource` will be created; otherwise, an instance of the default `UriSource` implementation will be created.
- New overloaded variant of `isAnnotated()` in `AnnotationSupport` that accepts `Optional<? extends AnnotatedElement>` instead of `AnnotatedElement`.
- New `--fail-if-no-tests` command-line option for the `ConsoleLauncher`.
    - When this option is enabled and no tests are discovered, the launcher will fail and exit with a status code of `2`.

JUnit Jupiter

Bug Fixes

- When using `@TestInstance(Lifecycle.PER_CLASS)` semantics, registered `AfterAllCallback` extensions are no longer invoked if an exception is thrown by the test class constructor. Consequently, `AfterAllCallback` extensions are now only invoked if `BeforeAllCallback` extensions are invoked.
- Test discovery no longer halts prematurely if a nested class or nested interface in a test class has an invalid class file.
- Certain categories of errors encountered during the test discovery phase no longer cause JUnit Jupiter to prematurely abort the entire discovery process.
    - Such errors are now logged, thereby enabling JUnit Jupiter to discover and execute as many tests as possible while still informing the user of containers and tests that could not be properly discovered.

New Features and Improvements

- Experimental support for parallel test execution. By default, tests are still executed sequentially; parallelism can be enabled using a configuration parameter (please refer to the User Guide for examples and configuration options).
- New support for the IBM AIX operating system in `@EnabledOnOs` and `@DisabledOnOs`.
- New `assertThrows` methods in `Assertions` provide a more specific failure message if the supplied lambda expression or method reference returns a result instead of throwing an exception.
- New `arguments()` static factory method in the `Arguments` interface that serves as an *alias* for `Arguments.of()`. `arguments()` is intended to be used via `import static`.
- New `get<Class>(index)` Kotlin extension method to make `ArgumentsAccessor` friendlier to use from Kotlin.
- New support for supplying a custom test source `URI` when creating a dynamic container or test.
    - See the new factory methods `dynamicContainer(String, URI, ⋯)` in `DynamicContainer` and `dynamicTest(String, URI, Executable)` in `DynamicTest` for details.

JUnit Vintage

No changes.

5.2.0

Date of Release: April 29, 2018

Scope: JUnit BOM, support for Maven Surefire 2.21.0 allowing builds with Java 9 and Java 10, *argument aggregation* and *widening primitive conversion* for arguments in parameterized tests, external factory methods for `@MethodSource`, as well as various minor improvements and bug fixes.

For complete details consult the 5.2.0 Release Notes online.