JUnit 5 Release Notes

Table of Contents

This document contains the *change log* for all JUnit 5 releases since 5.4 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.

5.5.0-RC1

Date of Release:

Scope:

For a complete list of all *closed* issues and pull requests for this release, consult the 5.5 RC1 milestone page in the JUnit repository on GitHub.

JUnit Platform

Bug Fixes

- A custom `ClassLoader` created for additional `--class-path` entries passed to the `ConsoleLauncher`

will now be closed after usage to gracefully free file handles.

Deprecations and Breaking Changes

- The internal `PreconditionViolationException` class in concealed package `org.junit.platform.commons.util` is now deprecated and has been replaced by an exception class with the same name in exported package `org.junit.platform.commons`.

New Features and Improvements

- `AnnotationSupport.findRepeatableAnnotations()` now finds repeatable annotations used as meta-annotations on other repeatable annotations.
- New `AnnotationSupport.findRepeatableAnnotations()` variant that accepts a `java.util.Optional<? extends AnnotatedElement>` argument.
- Exceptions thrown by `TestExecutionListeners` no longer cause test execution to abort. Instead, they will be logged as warnings now.
- New `MethodSource.from()` variant that accepts `String, String, Class<?>…` as arguments.

JUnit Jupiter

Bug Fixes

- Execution of dynamic tests registered via a `@TestFactory` method no longer results in an `OutOfMemoryError` if the executables in the dynamic tests retain references to objects consuming large amounts of memory. Technically speaking, JUnit Jupiter no longer retains references to instances of `DynamicTest` after they have been executed.

Deprecations and Breaking Changes

- Script-based condition APIs and their supporting implementations are deprecated with the intent to remove them in JUnit Jupiter 5.6. Users should instead rely on a combination of other built-in conditions or create and use a custom implementation of `ExecutionCondition` to evaluate the same conditions.

New Features and Improvements

- Support for declarative timeouts using `@Timeout` or configuration parameters (see User Guide for details)
- New overloaded variants of `Assertions.assertLinesMatch(…)` that accept a `String` or a `Supplier<String>` for a custom failure message.
- Failure messages for `Assertions.assertLinesMatch(…)` now emit each expected and actual line in a dedicated line.
- New Kotlin friendly `assertDoesNotThrow`, `assertTimeout`, and `assertTimeoutPreemptively` assertions have been added as top-level functions in the `org.junit.jupiter.api` package.
- New `emptyValue` attribute in `@CsvSource` and `@CsvFileSource`.

- Display names for test methods generated by the `ReplaceUnderscores` `DisplayNameGenerator` no longer include empty parentheses for test methods that do not declare any parameters.
- New `junit.jupiter.displayname.generator.default` configuration parameter to set the default `DisplayNameGenerator` that will be used unless `@DisplayName` or `@DisplayNameGeneration` is present.
- `MethodOrderer.Random` now generates a default random seed only once and prints it to the log in order to allow reproducible builds.
- Methods ordered with `MethodOrderer.Random` now execute using the `SAME_THREAD` concurrency mode instead of the `CONCURRENT` mode when no custom seed is provided.
- The declared field type for an extension registered via `@RegisterExtension` is no longer required to implement an `Extension` API. It is now sufficient if the extension implementation can be assigned to the declared field type. This provides extension authors greater flexibility as well as the ability to hide implementation details of the user facing extension API.
- All methods in the `TestWatcher` API are now interface `default` methods with empty implementations.
- New `InvocationInterceptor` extension API (see User Guide for details).
- A custom test source for a `DynamicContainer` or `DynamicTest` may now be a method URI — for example, `method:org.example.MyTestClass#myTestMethod()`.
- New `junit.jupiter.execution.parallel.mode.classes.default` configuration parameter allows to run top-level classes in parallel but their methods sequentially or vice versa (see User Guide for details).

JUnit Vintage

New Features and Improvements

- `junit:junit` is now a compile-scoped dependency of `junit-vintage-engine` to allow for easier dependency management in Maven POMs.
- Methods that are `public` are now preferred over other methods with the same name in the same test class when creating `MethodSources` for JUnit 4 `Descriptions`.

5.5.0-M1

Date of Release: March 19, 2019

Scope: Configurable test discovery implementation

For a complete list of all *closed* issues and pull requests for this release, consult the 5.5 M1 milestone page in the JUnit repository on GitHub.

JUnit Platform

New Features and Improvements

- Configurable test discovery implementation that can be reused by different test engines (see Javadoc of the `org.junit.platform.engine.support.discovery` package).
- New `isFinal()` and `isNotFinal()` methods in `ModifierSupport`.

JUnit Jupiter

New Features and Improvements

- Expected and actual values are now supplied for failed `boolean` assertions for enhanced IDE and reporting support — for example, when `assertTrue()` or `assertFalse()` fails.
- `@ValueSource` now additionally supports literal values of type `boolean` for parameterized tests.

JUnit Vintage

No changes.

5.4.2

Date of Release: April 7, 2019

Scope: Bug fixes since 5.4.1

For a complete list of all *closed* issues and pull requests for this release, consult the 5.4.2 milestone page in the JUnit repository on GitHub.

JUnit Platform

No changes.

JUnit Jupiter

Bug Fixes

- Parameterized tests no longer throw an `ArrayStoreException` when creating human-readable test names.

JUnit Vintage

Bug Fixes

- Safeguard against `Runners` that only report tests as failed but not as started or finished such as Spock in case of failures during data-provider preparation.

5.4.1

Date of Release: March 17, 2019

Scope: Bug fixes since 5.4.0

For a complete list of all *closed* issues and pull requests for this release, consult the 5.4.1 milestone page in the JUnit repository on GitHub.

Overall Improvements

- Fix `Specification-Version` entry in JAR manifests

JUnit Platform

Bug Fixes

- Restore compatibility with Android: Unsupported `Pattern` flags, like `UNICODE_CHARACTER_CLASS`, no longer cause class `StringUtils` to fail during initialization.

JUnit Jupiter

Bug Fixes

- Deletion of a temporary directory within a test no longer results in a test failure for a temporary directory supplied via `@TempDir`.

JUnit Vintage

Bug Fixes

- Fix reporting of finish events of intermediate containers with static and dynamic children, e.g. Spock test classes with regular and `@Unroll` feature methods in a test suite.

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 complete details consult the 5.4.0 Release Notes online.