JUnit 5 Release Notes

Table of Contents

This document contains the *change log* for all JUnit 5 releases since 5.6 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.

5.7.0-M1

Date of Release: April 19, 2020

Scope:

- New `@EnabledIf`/`@DisabledIf` annotations for conditional execution based on method calls
- New `MethodOrderer` named `DisplayName` that orders methods based on their display names
- New `DisplayNameGenerator` named `Simple` that removes parentheses for empty parameter lists
- `assertThrows()` for Kotlin can now be used with suspending functions
- `TestExecutionListener` deactivation via a configuration parameter
- `EngineTestKit` now allows for testing post-discovery filtering and pruning
- Improved interoperability with Spock for the Vintage test engine

For a complete list of all *closed* issues and pull requests for this release, consult the 5.7 M1 milestone page in the JUnit repository on GitHub.

JUnit Platform

Deprecations and Breaking Changes

- In the `EngineTestKit` API, the `all()`, `containers()`, and `tests()` methods in `EngineExecutionResults` that were deprecated in JUnit Platform 1.6.0 have been removed in favor of `allEvents()`, `containerEvents()`, and `testEvents()`, respectively.
- The following methods in `EngineTestKit` are now deprecated with replacements:
  - `execute(String, EngineDiscoveryRequest)` → `execute(String, LauncherDiscoveryRequest)`
  - `execute(TestEngine, EngineDiscoveryRequest)` → `execute(TestEngine, LauncherDiscoveryRequest)`
  - `Builder.filters(DiscoveryFilter…)` → `Builder.filters(Filter…)`

New Features and Improvements

- The number of containers and tests excluded by post discovery filters based on their tags is now logged, along with the exclusion reasons.
- New `junit.platform.execution.listeners.deactivate` configuration parameter that allows one to specify a comma-separated list of patterns for deactivating `TestExecutionListener` implementations registered via the `ServiceLoader` mechanism.
- The `@Testable` annotation may now be applied *directly* to fields.
- New `Node.DynamicTestExecutor#execute(TestDescriptor, EngineExecutionListener)` method for engines that wish to provide a custom `EngineExecutionListener` and cancel or wait for the execution of a submitted test via the returned `Future`.
- New `EngineExecutionListener.NOOP` `EngineExecutionListener` implementation.
- All declared methods in the `EngineExecutionListener` API now have empty `default` implementations.
- The `EngineTestKit` now reuses the same test discovery and execution logic as the `Launcher`. Thus, it's now possible to test an engine's behavior in the presence of post-discovery filters (e.g. tag filters) and with regard to pruning.
- The `EngineTestKit` now supports matching conditions with events loosely, i.e. an incomplete match with or without a fixed order.

JUnit Jupiter

Bug Fixes

- `@TempDir` is now able to clean up files in read-only directories.
- The Jupiter engine now ignores `MethodSelectors` for methods in non-Jupiter test classes instead of failing for missing methods in such cases.

New Features and Improvements

- New `@EnabledIf` and `@DisabledIf` annotations can be used to enable or disable a test or container based on condition methods.
- New `MethodOrderer` named `DisplayName` that sorts test methods alphanumerically based on their display names.
- New `DisplayNameGenerator` named `Simple` (based on `Standard`) that removes trailing parentheses for methods with no parameters.
- `assertThrows()` for Kotlin can now be used with suspending functions and other lambda contexts that require inlining.
- The `JRE` enum now provides a static `currentVersion()` method that returns the enum constant for the currently executing JRE, e.g. for use in custom execution conditions and other extensions.
- The `name` attribute of `@ParameterizedTest` is now clearly documented to be a `MessageFormat` pattern.
- Synthetic constructors are now ignored when instantiating a test class.
- The Javadoc for the `provideTestTemplateInvocationContexts()` method in `TestTemplateInvocationContextProvider` has been aligned with the actual implementation. Providers are now officially allowed to return an empty stream, and the error message when all provided streams are empty is now more helpful.
- New `getDisplayName()` method in `MethodDescriptor` for use in `MethodOrderer` implementations.

JUnit Vintage

Bug Fixes

- The Vintage engine no longer fails when resolving a `MethodSelector` for methods of test classes that cannot be found via reflection. This allows selecting Spock feature methods by their source code name even though they have a generated method name in the bytecode.

New Features and Improvements

- The internal `JUnit4VersionCheck` class — which verifies that a supported version of JUnit 4 is on the classpath — now implements a lenient version ID parsing algorithm in order to support custom version ID formats such as `4.12.0`, `4.12-patch_1`, etc.

5.6.2

Date of Release: April 10, 2020

Scope: Bug fixes since 5.6.1

For a complete list of all *closed* issues and pull requests for this release, consult the 5.6.2 milestone page in the JUnit repository on GitHub.

JUnit Platform

Bug Fixes

- `ReflectionSupport.findNestedClasses()` no longer detects inner class cycles for classes that do not match the supplied `Predicate`. For example, JUnit Jupiter no longer throws an exception if an inner class cycle is detected in a nested class hierarchy whose inner classes are not annotated with `@Nested`.

JUnit Jupiter

Bug Fixes

- Test discovery no longer halts with an exception for inner class hierarchies that form a cycle if such inner classes are not annotated with `@Nested`.

JUnit Vintage

Bug Fixes

- Generating display names from JUnit 4 `Descriptions` now falls back to `getDisplayName` if `getMethodName` returns a blank `String` instead of throwing an exception.

5.6.1

Date of Release: March 22, 2020

Scope: Bug fixes since 5.6.0

For a complete list of all *closed* issues and pull requests for this release, consult the 5.6.1 milestone page in the JUnit repository on GitHub.

JUnit Platform

Bug Fixes

- In order to avoid file locking issues on Microsoft Windows, URLs are no longer cached when loading `junit-platform.properties` files.
- The presence of multiple `junit-platform.properties` files on the classpath now only results in a warning if the files have different URLs.

JUnit Jupiter

Bug Fixes

- `TestInstancePreDestroyCallback` extensions are now invoked in reverse registration order when `PER_CLASS` test instance lifecycle semantics have been configured.

JUnit Vintage

No changes.

5.6.0

Date of Release: January 20, 2020

Scope:

- New `@EnabledForJreRange` and `@DisabledForJreRange` execution conditions
- `@Order` allows to specify relative order
- Parameter names are included in default display names of parameterized test invocations
- Improvements to `@CsvSource` and `@CsvFileSource`
- New `TestInstancePreDestroyCallback` extension API
- Performance improvements and bug fixes for the Vintage engine
- Improved error reporting for failures during test discovery and execution
- Support for using `any()` and `none()` in tag expressions
- `org.junit.platform.console` now provides a `java.util.spi.ToolProvider`
- `DiscoverySelectors` for tests in inherited nested classes
- OSGi metadata
- Minor bug fixes and improvements

For complete details consult the 5.6.0 Release Notes online.