How to migrate to JUnit 5

This guide should help you as a module developer to migrate your tests to JUnit 5. If you are not yet convinced that you should migrate read on. If you are jump straight to the next section.

WORK IN PROGRESS

We are still working on the migration of openmrs-core 💚 TRUNK-5811 - Update Junit from 4 to 5 🟢 IN PROGRESS. JUnit 5 tests will be supported from the release of 2.4.0 on. Your JUnit 4 tests will still run without any change from your side.

You will only be able to migrate all your tests once the above task is marked as done.

So the information here is preliminary and might also change.

Why should I migrate?

We are in the process of migrating openmrs-core tests from JUnit 4 to JUnit 5 💚 TRUNK-5811 - Update Junit from 4 to 5 🟢 IN PROGRESS. JUnit 5 has been released on Sep 10, 2017. JUnit 4 is still maintained and thus receives updates. At some point it will have reached its end of life. New developers with some Java experience coming to OpenMRS from other projects will very likely have written tests in JUnit 5. Since JUnit 5 is the current version and out since Sep 2017. JUnit 5 offers a lot of interesting new feature that we can leverage. An incomplete list of interesting features

- **conditional test execution** (based on environment variables, JDK version, ...). This would allow us for example to run integration tests against mysql or postgresql in an environment where they are actually deployed
- **tags** could be used for tagging tests as integration tests regardless of what their name is. maven-surefire could then use them to filter unit vs integration tests
- **parametrized tests** can be written next to standard tests
- the **new extension model** allows us to potentially write our own extensions that can be used alongside Spring/Mockito extensions (extensions replacing the runners (in JUnit 4 only 1 runner per class is allowed) and rules)
- **timeouts**
- ...

What do I need to do?

Context

The JUnit team has provided great support for teams to gradually migrate over to JUnit 5. When updating openmrs-core our goal was to provide the same to you as module developers.

This means that once you update your dependency on openmrs-core (TODO and most likely also openmrs-test) to 2.4.x you will be able to

- still run your existing JUnit 4 tests (including the tests depending on the BaseModuleContextSensitiveTest.java and BaseModuleWebContextSensitiveTest.java) and migrate tests one after the other (or all at once if you prefer) over to JUnit 5

If you are interested in some background information on the migration from the JUnit team see https://junit.org/junit5/docs/current/user-guide/#migrating-from-junit4

Tools

We used IntelliJ IDEA 2020.1.3 (Community Edition) since it provides a migration tool that automates most steps.

Steps

2. Replace all usages of ExpectedException or @Test (expected = APIException.class) with the assertThrows JUnit 4 method. Try using IntelliJ Structural search & replace within the entire project. See discussion on talk about example usages. Read more on the why at https://issues.openmrs.org/browse/TRUNK-5813
3. Update your dependency on openmrs-core (most likely also openmrs-test) to 2.4.x. you can use mvn dependency:tree to display the project dependencies in tree format. Make sure these dependencies are present.
3. Make sure you pulled the latest changes of the OpenMRS module you want to migrate.

4. Ensure IntelliJ Settings Auto Import are configured as shown below.

```
org.openers.test:openmrs-test:2.2.0-SNAPSHOT
+ org.springframework:spring-test:2.0.8.RELEASE:test
+ com.h2database:h2:1.4.208:dist
+ junit:junit:junit-4.8.2:jar
| org.junit.jupiter:junit-jupiter-api:5.6.2:jar
| org.junit.jupiter:junit-jupiter-engine:5.6.2:jar
| org.junit.jupiter:junit-platform-commons:1.6.2:jar
| org.junit.jupiter:junit-platform-engine:1.6.2:jar
```

5. Ensure IntelliJ Settings Editor Code Style Java are setup as explained here (Developer How-To Setup And Use IntelliJ - Java Imports) so you do not use imports with wildcards (*)

1. Open IntelliJ and select Refactor Migrate.
2. Select the JUnit (4.x  5.0) migration as shown below.
1. Edit the migration map so it looks like

1. Click ok and run
2. Look at the changes and see if they make sense to you. Try to run a single unit and integration test to see if they are still passing. Run all tests to see if they are still passing.
3. Commit and push your changes. It’s up to you if you want to have a big commit with all the changes or create more commits only containing a few files.

If you want to see a video of how the migration feature works see