ToiletPaper #125

Best Practice RxJava2 Testing

Author: Alexander Pöhlmann / Software Engineer / Office Leipzig

✗ Problem

Many may wonder how to test correctly with RxJava, because it is not so easy to test an asynchronous process. In many cases, the other thread is slower or faster than expected. This leads to problems when writing tests with the usual JUnit functionalities –tests are often paused too long or fail from time to time.
Another problem is that some tests often take very long to complete when executed normally. For this reason, tests are often written incompletely or not written at all.

✓ Solution

To avoid this, RxJava offers several possibilities to test these processes optimally and efficiently. But keep in mind that the naming has changed from RxJava1 to RxJava2 (maybe also RxJava3). However, the basic logic has stayed the same.
On the one hand, there is the ***TestObserver*** and the ***TestSubscriber***, which register to the events from the respective ***ObservableSource***. They have different **assert** methods to check the received values. But the **await** methods are very important as well. With these, tests can be paused and only when the conditions or timeouts are fulfilled, the tests will be continued.
For many processes, the **await** methods are sufficient, but sometimes periods of several hundred milliseconds or more have to be bridged. To wait the whole runtime would be very impractical and have a serious effect on the runtime of the tests after a certain time.
RxJava2 provides the ***Testscheduler*** for this. It allows to fast-forward RxJava operations. Important to know: the ***Testscheduler*** does not work by itself and operations are continued only when using the fast-forward function.

➔ Tools and Terms

- TestSubscriber
  - Allows the examination of events using the various **assert*** methods.
  - Created by the ***Flowable***::test method or registered classically via ***Flowable***::subscribe.
- TestObserver
  - Allows the examination of events using the various **assert*** methods.
  - Created by the ***Observable/Single/Completable***::test method or classically registered via ***Observable/Single/Completable***::subscribe.
- TestScheduler
  - ***TestScheduler*** is able to manipulate time by fast-forwarding certain time intervals.
  - Very useful for processes that are only triggered at certain times.
- await* methods
  - Test thread is paused until the respective condition is fulfilled.
  - There are different **await*** methods:
    - **await**() – pauses the thread for a certain amount of time (Is the same as Thread::sleep)
    - **awaitCount**() – waits until the number of events is reached.
    - **awaitDone**()/**awaitTerminalEvent**() – waits until the source is completed (even in case of errors).
- assert* methods
  - Various methods for testing the received values.
  - Behaves similar to the testing methods of the org.junit.Assert class.
  - Examples: **assertComplete**(), **assertEmpty**(), **assertError**(), **assertNever**(), **assertNoErrors**(), **assertNoValues()**.

+ Further Aspects

- https://www.baeldung.com/rxjava-testing
- https://medium.com/@vanniktech/testing-rxjava-code-made-easy-4cc32450fc9a