ToiletPaper #107

Better testing, better life – with AssertJ

Author: Alexandros Panagiotidis / Senior Software Architect / Office Stuttgart

✕ Problem

How many variants do you know to check whether a collection is empty or not empty in JUnit? You certainly thought of `assertTrue(foo.isEmpty())`. But have you ever come across `assertEquals(0, foo.size())` or `assertFalse(foo.size() > 0)`? Now think about how many ways there are to compare the contents of one collection with another. This is even more confusing when the order of the elements plays a role (or not!).

If your head isn't about to explode yet: consider comparing objects if `equals()` has not been implemented or has been implemented incorrectly. All of this and more can be done easily and elegantly with an expressive framework - e.g. AssertJ.

➔ Solution and Examples

Thanks to Fluent interfaces, AssertJ allows you to chain your expectations to check multiple properties easily:

```
assertThat(jambit.getMission()).isNotNull().isEqualTo("100% Begeisterung");
```

Assertions of typed objects directly eliminate some error classes when creating tests. For example, the size and contents of collections can be checked easily:

```
assertThat(Arrays.asList("Where", "innovation", "works")).hasSize(3)
    .containsExactly("Where", "innovation", "works") // Order is important
    .containsOnlyOnce("works", "Where", "innovation"); // Order is unimportant
```

Furthermore, checking predicates for a whole collection is trivial:

```
assertThat(jambit).extracting(Jambit::getEmployees)
    .allMatch(JambitEmployee::isTopOfMind); // alternatively {any,none}Match
```

Individual fields and results of methods can be verified easily with `extracting()`:

```
assertThat(jambit)
    .extracting(Jambit::getBosses).extracting("nickname", JambitBoss::getFullName)
    .containsOnly(tuple("Felli", "Peter F. Fellinger"), tuple("Harti", "Markus
Hartinger"));
```

As you can see, field names can also be used instead of accessors. In addition to individual values, you can also compare objects where `equals()` is missing:

```
assertThat(jambitStuttgart).extracting(JambitLocation::getAddress)
    .isEqualToComparingFieldByFieldRecursively(JAMBIT_STUTTGART_EXPECTED_ADDRESS);
```

For exceptions, there are also some useful matchers:

```
assertThatThrownBy(() -> throw DeveloperException("Out of coffee"))
    .isInstanceOf(DeveloperException.class).hasMessage("Out of coffee");
```

Finally, here is a more complex example where a collection is compared pairwise to a second one and all assertions are evaluated for each pair:

```
assertThat(jambit).extracting(getDivisions)
    .zipSatisfy(fetchDivisionNames(), (JambitDivision actualDivision, String
expectedDivisionName) -> {
        try (AutoCloseableSoftAssertions softly = new
AutoCloseableSoftAssertions()) {

softly.assertThat(actualDivision).extracting("name").startsWith(expectedDivisionNa
me);

softly.assertThat(calculatePowerLevel(actualDivision)).isGreaterThan(9000);
        }
    });
```

✚ Further Aspects

As you can see, AssertJ has a lot to offer. It's best to have a look at the feature highlights and get started right away. There are also some migration guides in the documentation – but you can also use AssertJ together with other frameworks.