Better Java Asserts

Toby Fleming

April 19, 2020

What does a bad assert look like?

Facile example

```
import static org.junit.Assert.assertTrue;
assertTrue(a.equalsIgnoreCase(b));
```

```
java.lang.AssertionError
    at org.junit.Assert.fail(Assert.java:86)
    ...
```

😭😭😭

Different example

realistic example:

```
final Result a = new API().getResult();
final Result b = new Result.Builder().build();
assertEquals(a, b);
```

simplified example:

```
assertEquals("foo", "bar");
```

Pop Quiz

What’s the assert message?

```
assertEquals("foo", "bar");
```

What’s the assert message?

```
assertEquals("foo", "bar");
```

a) `expected: <foo> but was: <bar>`

What’s the assert message?

```
assertEquals("foo", "bar");
```

a) `expected: <foo> but was: <bar>`

b) `expected: <bar> but was: <foo>`

What’s the assert message?

```
assertEquals("foo", "bar");
```

a) `expected: <foo> but was: <bar>`
b) `expected: <bar> but was: <foo>`
c) all of the above?

What's the assert message?

```java
assertEquals("foo", "bar");
```

a) `expected: <foo> but was: <bar>`
b) `expected: <bar> but was: <foo>`
c) all of the above?
d) none of the above

The fact you had to think about it means `assertEquals()` is bad API design

The fact you had to think about it means `assertEquals()` is bad API design

```
public static void assertEquals(Object expected, Object actual)
```

(`https://junit.org/junit4/javadoc/4.12/org/junit/Assert.html`)

What does a good assert look like?

AssertJ > JUnit

JUnit:

```
assertEquals(expected, actual);
```

AssertJ:

```
assertThat(actual).isEqualTo(expected);
```

(okay, the order still doesn’t seem super obvious, but will be as soon as we start using other assertions)

AssertJ > JUnit

- obvious argument order
- strongly typed; can’t accidentally compare incompatible types
- AssertJ assertions are chainable, very powerful, and extremely informative
- you can write your own assertion helpers
- better NPE information!
- AssertJ seems more to type, but we’ll get to that...

Still not convinced?

isEqualToIgnoringCase

```
assertTrue("foo".equalsIgnoreCase("bar"));

...
```

```
assertThat("foo").isEqualToIgnoringCase("bar");

Expecting:
 <"foo">
to be equal to:
 <"bar">
ignoring case considerations
```

hasSize

```java
final List<String> list = ImmutableList.of("foo");
assertThat(list).hasSize(2);
```

```
java.lang.AssertionError:
Expected size:<2> but was:<1> in:
<["foo"]>
```

you can actually see what’s in the list!

NullPointerException

```java
final List<String> list = null;
assertThat(list).hasSize(2);
```

```
java.lang.AssertionError:
Expecting actual not to be null
```

it’s a small win this isn’t an NPE; but this does make triage easier.

More complex asserts!

```
assertThat(string)
    .startsWith("foo")
    .endsWith("bar");

assertThat(list)
    .hasSize(7)
    .contains(item);

assertThatExceptionOfType(ArithmeticException.class)
    .isThrownBy(() -> { ... })
    .hasMessageContaining("foo")
```

Other learnings

JSONAssert

- what about `org.skyscreamer.jsonassert.JSONAssert`?
- can be good, can print absolutely useless errors in rare cases
- hand-rolled JSON strings in Java *suck* because of escaping; brittle
- old version were worse printing no useful error

JSONAssert

- what about `org.skyscreamer.jsonassert.JSONAssert`?
- can be good, can print absolutely useless errors in rare cases
- hand-rolled JSON strings in Java *suck* because of escaping; brittle
- old version were worse printing no useful error

```
JSONAssert.assertEquals("{\"EXPECTED\": 1}", "\"ACTUAL\"", true);
```

```
Expected: a JSON object
     got: org.skyscreamer.jsonassert.JSONParser$1@5d16055
```

(okay, contrived, but still interesting)

TL;DR

TL;DR

- AssertJ is really nice
- obvious argument order
- strongly typed
- better NPE handling
- easier to write complex asserts, more specific tests, less brittle
- richer assertion messages means less time debugging, grok the problem quicker
- prefer AssertJ over Hamcrest, because auto-complete makes discovering assertions easy, less nesting due to fluent style

Fin