Mockito how does it work?

Background

So what is Mockito? It is framework that is used on top of testing framework library like JUnit test to provide mocking and stubbing capability of objects for your unit test or integration tests.

"Mocking is the act of removing external dependencies from a unit test in order to create a controlled environment around it. Typically, we mock all other classes that interact with the class that we want to test". This is so that you can get a consistent behavior regardless how the external dependency actually react.

So we are basically pulling out the external dependency and substitute in a fake object in it's place to use for testing purposes only.

Why do you need to mock (fake) an object? Well, if the external dependency that your code base depends on is slow, and requires reading large files, or needs a database connection setup, you wouldn't want to set that up every time your test is run right?

That's where mocking comes into play, you assume that the external dependency you use are working, then you will fake those method calls that your code base is calling on those external dependency to return values that you expect it to return.

Mock? Stub?

In mocking theory:

- A stub is a fake class that comes with preprogrammed return values. This gives you full control of the object, letting you control the return value when a certain method is called.
- A mock is a fake class as well, that can be examined after the test is finished for its interaction with the class under test. You can whether a particular method is invoked and also see how many times that method has been invoked.

> So stubbing is the act of faking a method. You choose how the method behaves. This is done using when().thenReturn() calls.
> Mocking on the other hand is done by using verify() from the Mockito library, to inspect the mocked object.

In Mockito, it uses the terminology "mock "for both stub and a mock.

Basic stubbing with mockito

Stubbing is done via the `when().thenReturn()` method call on the mocked object. For example:

```
when(entityManager.find(2)).thenReturn(1);
when(entityManager.find(anyString())).thenReturn(5);
```

When the `find` method is called on `entityManager` and passed in the int value 2, then it will return 1.

When the `find` method is called on `entityManager` and passed in any string value, then it will return 5.

Assuming that the find method is overloaded of course.

What happens if you don't stub method and call it?

By default, for all methods that return values, if it returns an object it will return null, an empty collection, or an appropriate primitive default value if the method returns a primitive such as 0 for integers, false for booleans and so on.

This means that a mocked object doesn't inherit any of the original method that the class actually has!

You can change this behavior to have the real methods from the class to be called when not stubbed by adding

```
mock(Example.class, Mockito.CALLS_REAL_METHODS);
```

However, this is not recommended since they could use fields that doesn't exist in the mocked object :).

Basic mocking with mockito

Now we discussed stubbing how about mocking? How do we inspect whether or not the method is called in an object and how many it has been called?

We can do that by calling the verify function.

```
verify(emailSender).sendEmail(sampleCustomer);
verify(emailSender, times(0)).sendEmail(sampleCustomer);
```

The first verify will check whether or not the object `emailSender` called `sendEmail` with the specified parameter once. By default if you don't provide the number of invocation to check it will default to

1. i.e. it should only be invoked once.

The second verify will check that the object didn't call `sendEmail` at all.

Revision #1
Created 2023-04-14 20:22:11 UTC by Tamarine
Updated 2023-04-14 21:46:13 UTC by Tamarine