PowerMockito mock single static method and return object

72

I want to mock a static method m1 from a class which contains 2 static methods, m1 and m2. And I want the method m1 to return an object.

I tried the following

1)

```
PowerMockito.mockStatic(Static.class, new Answer<Long>() {
         @Override
         public Long answer(InvocationOnMock invocation) throws Throwable {
            return 1000l;
         }
      });
```

This is calling both m1 and m2, which has a different return type, so it gives a return type mismatch error.

2) `PowerMockito.when(Static.m1(param1, param2)).thenReturn(1000l);` But this is not called when m1 is executed.

3) `PowerMockito.mockPartial(Static.class, "m1");` Gives compiler error that mockPartial not available, which I got from http://code.google.com/p/powermock/wiki/MockitoUsage.

asked May 14 '12 at 12:19

user1393653
361133

1 Answer

×
105

What you want to do is a combination of part of 1 and all of 2.

You need to use the PowerMockito.mockStatic to **enable** static mocking for all static methods of a class. This means make it **possible** to stub them using the when-thenReturn syntax.

But the 2-argument overload of mockStatic you are using supplies a default strategy for what Mockito/PowerMock should do when you call a method you haven't explicitly stubbed on the mock instance.

From the javadoc:

> Creates class mock with a specified strategy for its answers to interactions. It's quite advanced feature and typically you don't need it to write decent tests. However it can be helpful when working with legacy systems. It is the default answer so it will be used only when you don't stub the method call.

The **default** default stubbing strategy is to just return null, 0 or false for object, number and boolean valued methods. By using the 2-arg overload, you're saying "No, no, no, by default use this Answer subclass' answer method to get a default value. It returns a Long, so if you have static methods which return something incompatible with Long, there is a problem.

Instead, use the 1-arg version of mockStatic to enable stubbing of static methods, then use when-thenReturn to specify what to do for a particular method. For example:

```
import static org.mockito.Mockito.*;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.invocation.InvocationOnMock;
import org.mockito.stubbing.Answer;
import org.powermock.api.mockito.PowerMockito;
import org.powermock.core.classloader.annotations.PrepareForTest;
import org.powermock.modules.junit4.PowerMockRunner;

class ClassWithStatics {
  public static String getString() {
    return "String";
  }

  public static int getInt() {
    return 1;
  }
}

@RunWith(PowerMockRunner.class)
@PrepareForTest(ClassWithStatics.class)
public class StubJustOneStatic {
  @Test
  public void test() {
    PowerMockito.mockStatic(ClassWithStatics.class);

    when(ClassWithStatics.getString()).thenReturn("Hello!");

    System.out.println("String: " + ClassWithStatics.getString());
    System.out.println("Int: " + ClassWithStatics.getInt());
  }
}
```

The String-valued static method is stubbed to return "Hello!", while the int-valued static method uses the default stubbing, returning 0.

answered May 18 '12 at 12:21

Tom Tresansky
10.2k1467111