Verify static method was called with PowerMock

automationrhapsody.com/verify-static-method-called-powermock

By Lyudmil Latinov May 25, 2017

Post summary: How to verify that static method was called during a unit test with PowerMock.

This post is part of PowerMock series examples. The code shown in examples below is available in GitHub java-samples/junit repository.

In Mock static methods in JUnit with PowerMock example post, I have given information about PowerMock and how to mock a static method. In the current post, I will demonstrate how to verify given static method was called during execution of a unit test.

Example class for unit test

We are going to unit test a class called ***LocatorService*** that internally uses a static method from utility class ***Utils***. Method ***randomDistance(int distance)*** in ***Utils*** is returning random variable, hence it has no predictable behavior and the only way to test it is by mocking it:

```
public class LocatorService {
public Point generatePointWithinDistance(Point point, int distance) {
return new Point(point.getX() + Utils.randomDistance(distance),
point.getY() + Utils.randomDistance(distance));
}
}
```

And Utils class is:

```
import java.util.Random;
public final class Utils {
private static final Random RAND = new Random();
private Utils() {
}
public static int randomDistance( int distance) {
return RAND.nextInt(distance + distance) - distance;
}
}
```

Nota bene: it is good code design practice to make utility classes final and with a private constructor.

Verify static method call

This is the full code. Additional details are shown below it.

```
package com.automationrhapsody.junit;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.internal.verification.VerificationModeFactory;
import org.powermock.api.mockito.PowerMockito;
import org.powermock.core.classloader.annotations.PrepareForTest;
import org.powermock.modules.junit4.PowerMockRunner;
@RunWith (PowerMockRunner. class )
@PrepareForTest (Utils. class )
public class LocatorServiceTest {
private LocatorService locatorServiceUnderTest;
@Before
public void setUp() {
PowerMockito.mockStatic(Utils. class );
locatorServiceUnderTest = new LocatorService();
}
@Test
public void testStaticMethodCall() {
locatorServiceUnderTest
.generatePointWithinDistance( new Point( 11 , 11 ), 1 );
locatorServiceUnderTest
.generatePointWithinDistance( new Point( 11 , 11 ), 234 );
PowerMockito.verifyStatic(VerificationModeFactory.times( 2 ));
Utils.randomDistance( 1 );
PowerMockito.verifyStatic(VerificationModeFactory.times( 2 ));
Utils.randomDistance( 234 );
PowerMockito.verifyNoMoreInteractions(Utils. class );
}
}
```

Explanation

Class containing static method should be prepared for mocking with***PowerMockito.mockStatic(Utils.class)*** code. Then call to static method is done inside ***locatorServiceUnderTest .generatePointWithinDistance()*** method. In this test, it is intentionally called 2 times with different distance (1 and 234) in order to show the verification which consists of two parts. First part is ***PowerMockito.verifyStatic(VerificationModeFactory.times(2))*** which tells PowerMock to verify static method was called 2 times. The second part is ***Utils.randomDistance(1)*** which tells exactly which static method should be verified. Instead of ***1*** in the brackets you can use***anyInt()*** or ***anyObject()***. ***1*** is used to make verification explicit. As you can see there is second verification that ***randomDistance()*** method was called with ***234*** as well: ***PowerMockito.verifyStatic(VerificationModeFactory.times(2)); Utils.randomDistance(234);***.

Conclusion

PowerMock provides additional power to Mockito mocking library which is described in Mock JUnit tests with Mockito example post. In the current post, I have shown how to verify static method was called. It is very specific as verification actually consists of two steps.

Related Posts

- PowerMock examples and why better not to use them
- Mock static methods in JUnit with PowerMock example
- Mock JUnit tests with Mockito example

Category: Java, Unit testing | Tags: JUnit, PowerMock