PowerMock examples and why better not to use them

automationrhapsody.com/powermock-examples-better-not-use

By Lyudmil Latinov May 31, 2017

Post summary: In this post, I have summarised all PowerMock examples I've given so far. More important I will try to give some justification why I think necessity to use PowerMock is considered an indicator for a bad code design.

All code examples are available in GitHub java-samples/junit repository.

PowerMock

PowerMock is a framework that extends other mock libraries giving them more powerful capabilities. PowerMock uses a custom classloader and bytecode manipulation to enable mocking of static methods, constructors, final classes and methods, private methods, removal of static initializers and more.

PowerMock series

So far in my blog, I have written a lot for PowerMock. Even more than I have written for Mockito which actually deserves better attention. Post from PowerMock series are:

- Mock static methods in JUnit with PowerMock example
- Verify static method was called with PowerMock
- Call private method with PowerMock
- Mock private method call with PowerMock
- Mock new object creation with PowerMock

Why avoid PowerMock

I have worked on a project where PowerMock was not needed at all. We had 91.6% code coverage only with Mockito. Initially, it was 85% but when we utilized PITest we increased the code coverage. See more on PITest in Mutation testing for Java with PITest post. I also have worked on an old product where without PowerMock you cannot do decent unit testing. PowerMock was a must in order to achieve our goal of 80% code coverage. I can easily compare those two projects. The old one had large classes with lots of private methods and used lots of static methods. It was really hard to maintain that code. In this post, I'm not going to talk about SOLID because I do not consider myself a total expert on the subject. There are lots of discussions over the internet about pros and cons of static methods so everyone can decide personally. For me, I've come to a conclusion that necessity of using PowerMock in a project is an indicator for bad code design. In later projects, PowerMock is not used at all. If something cannot be unit tested with Mockito then the class is refactored.

How to avoid PowerMock

PowerMock features described here are related to static methods, public methods and creating new objects.

Mock or verify static methods

I’m not saying don’t use static methods, but they should be deterministic and not very complex. Not being able to verify static method was called is a little pain but most important is input and output of your method under test, what internal call it is doing is not that important.

Mock or call private methods

Private methods are not supposed to be tested at all. It is like they do not exist. If a class is complex enough so you have to call private methods or to test individually private methods then this class might be good to be split up.

Mock new object creation

Instead of creating the object in the class use dependency injection to provide it to the class from outside either via a constructor or via a setter. This was you can very easily test this class by injecting a mock.

Conclusion

This is a very controversial post. On one hand, I describe how to use PowerMock and what features it has, on the other hand, I state that you’d better not use it. PowerMock is extremely powerful and can do almost anything you need in your testing, but for me, the necessity of using PowerMock is an indicator of bad code design.

Related Posts

- Mock new object creation with PowerMock
- Mock private method call with PowerMock
- Call private method with PowerMock
- Verify static method was called with PowerMock
- Mock static methods in JUnit with PowerMock example
- Mock JUnit tests with Mockito example

Category: Java, Unit testing | Tags: PowerMock