

Advanced Mocking Techniques



Nicolae Caprarescu

FULL-STACK SOFTWARE DEVELOPMENT CONSULTANT

www.properjava.com



Overview



More mock verification

Partial mocks

Dealing with code that is difficult to test



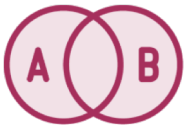
More Verification Tools



`verifyNoMoreInteractions(mock)`



Called stubbed methods are also treated as interactions!



`verifyNoMoreInteractions(ignoreStubs(mock))` ignores stubs for the purposes of verification

Mockito Strictness: Default Modes



Mockito 2.X: WARN

Complaints logged in console when things are not ideal



Mockito 3: STRICT_STUBS (*planned*)

Enabling this mode explicitly in Mockito 2.X means `ignoreStubs(...)` is not necessary anymore

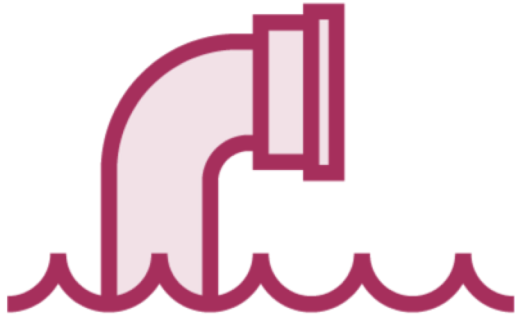
Demo



Verify no unexpected invocations on your
mocks



Partial Mocks



You spy on **real** objects



Use occasionally



Spies are partial
mocks



When creating a
spy, Mockito
creates a copy
of the object, it
doesn't
delegate calls



Demo



Creating spies

Using spies to test tricky situations



Spies should only be used occasionally and come in useful when you find yourself in a situation with code that impossible, or too costly, to change



Solution One: The PowerMock Framework



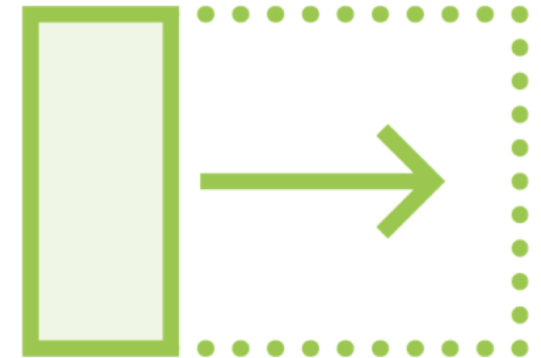
Mockito cannot mock static methods, such as `LocalDate.now()`



This solution is useful when you *can't* change the source code making static calls



<https://github.com/powermock/powermock/wiki/Mockito>



We won't cover PowerMock in this course



Solution Two: Java's Clock Class



Pluggable
abstraction of
time which
simplifies tests



This solution is
useful when you
can change the
source code
making static
calls



<https://docs.oracle.com/javase/8/docs/api/java/time/Clock.html>
! (since Java 1.8)



Let's see this in
action!



Demo



Testing code that deals with time



Mocking Final Methods



Mocking final classes and methods is now supported (since 2.1.0)



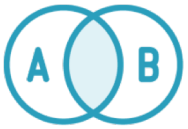
Final methods/classes can be mocked just like non-final ones



`‘/mockito-extensions/org.mockito.plugins.MockMaker’` containing a single line: `‘mock-maker-inline’`



Selective Partial Mocks



You can turn a mock into a spy by selectively enabling real methods



```
SomeType spy = spy(SomeType.class);
```

```
SomeType mock = mock(SomeType.class);
```

```
when(mock.someMethod).thenCallRealMethod();
```



Summary



More verification toolkit

- verifyNoMoreInteractions
- verifyNoMoreInteractions(ignoreStubs (...))

Partial mocks (spies)

Tested tricky scenarios

