Tasty Mocking Framework

TDD – Mockito *Part 1*

Jerry Wang

Software Developer

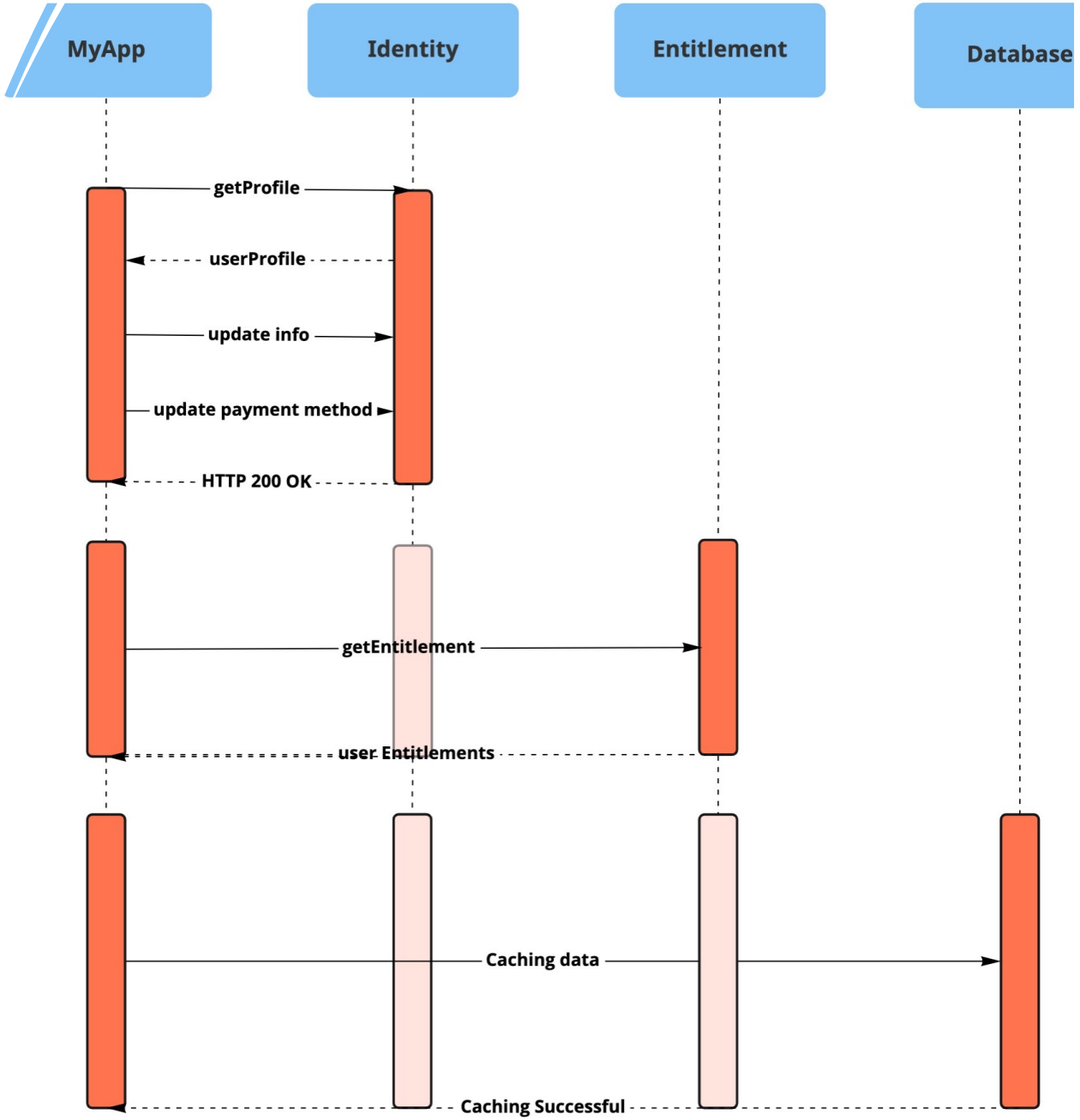
Feb 2023

Our Agenda

- A few words on the TDD
- Mocking
- Test Doubles
- Mockito in the sense of BDD
- Popular mocking concepts
- Live coding
- Final thoughts
- Questions

How do you go about delivering this requirement?

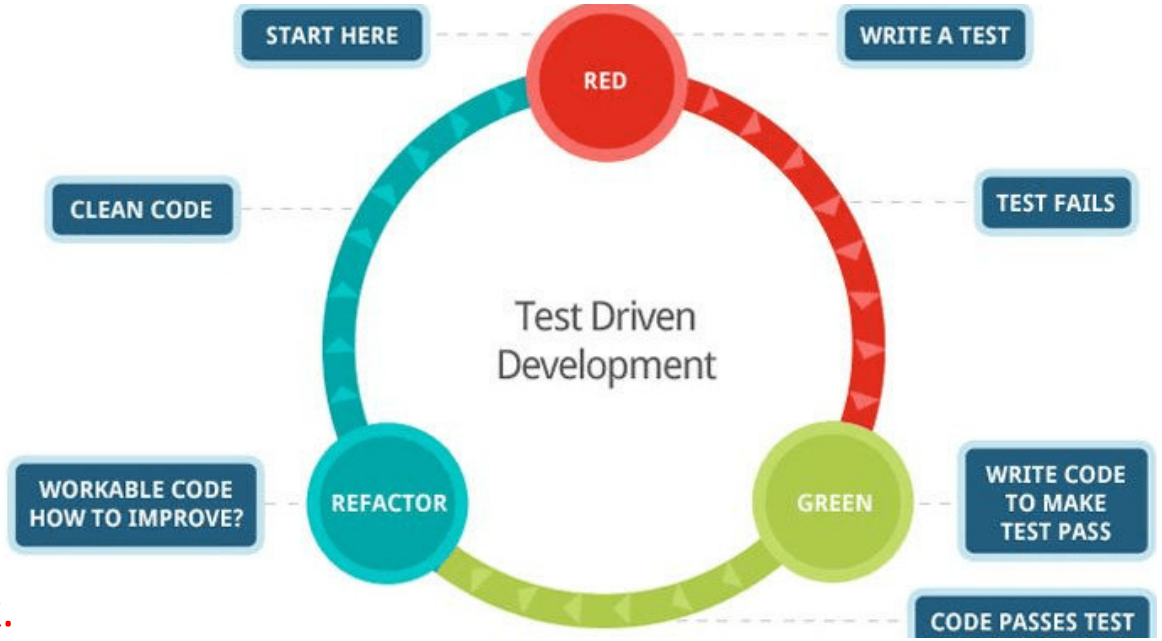
- What's the first step ?
- How do you guarantee minimum defects?
- Do you see any concern in this diagram?
- What will the codebase look like in a year of time?



Test Driven Development (TDD)

The Good

- Makes code easier to maintain and refactor.
- Makes collaboration easier and more efficient, team members can edit each others code with confidence
- Helps prevents defects
- Helps programmers really understand their code
- Clarity, Clarity, Clarity

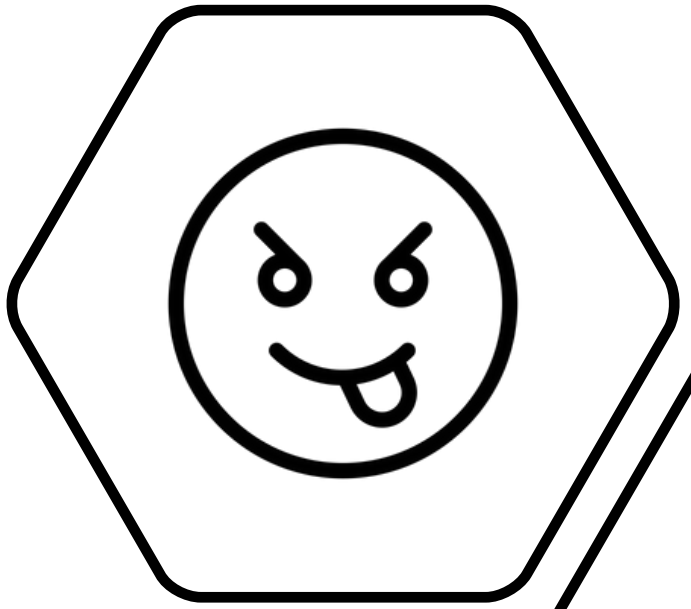


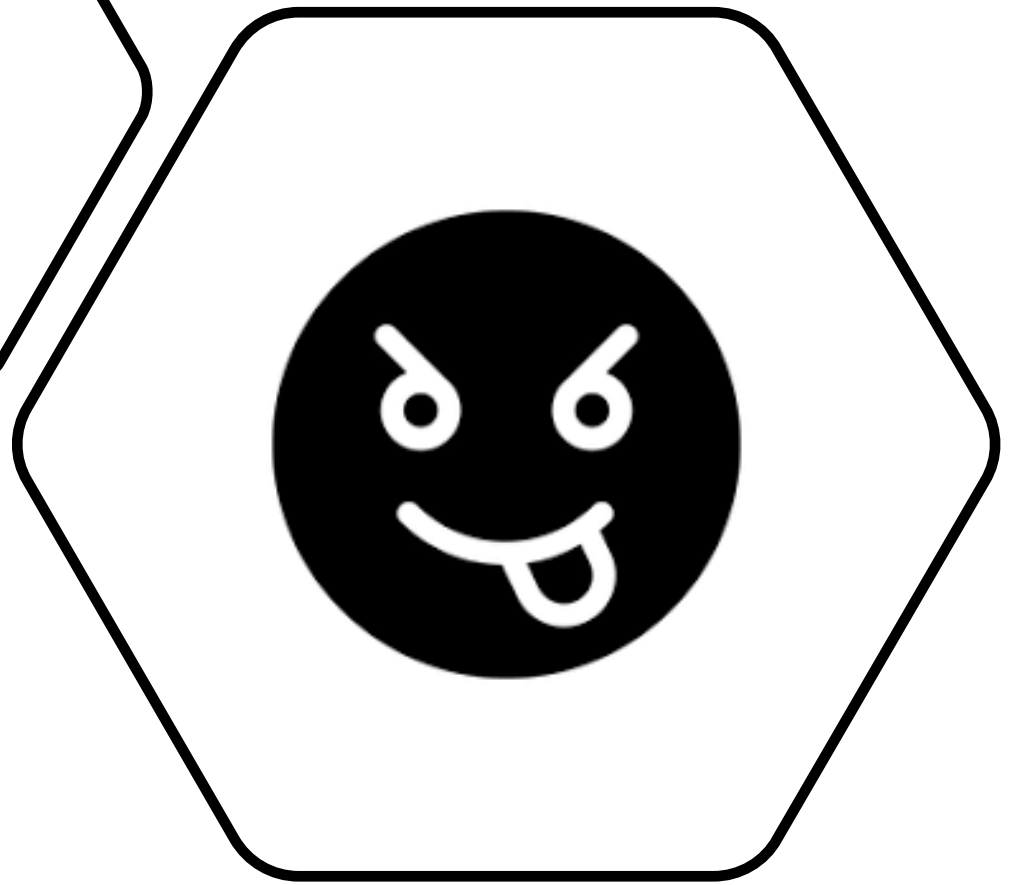
The Bad

- The test suite itself has to be maintained
- Slows down development, Initially
- Mock a lot of things or things that are difficult to mock.

The Verdict

- It’s beneficial in the long term, but painful right now





Mocking

- Allows you to focus on the unit you've trying to test by replacing the unit's real dependencies with tests-only collaborators.
- This allows you to reason about the unit in isolation without having to deal with the rest of code base at the same time.

Why Use Mocking?

- Eliminate non-determinism and randomness
- Reduces complexity – increase flexibility
- Improve test execution
  - Speed
  - Reliability
- Support collaboration

Test Doubles

“Double is a generic term for any case where you replace a production object for testing purposes”

Martin Fowler

Stub

Mock

Spy

https://martinfowler.com/bliki/TestDouble.html

Test Doubles: Stub

- Provides ‘canned’ answers
- Not intelligent enough to response with anything else

Stub

Test Doubles: Mock

- Uses expectations
- Can fail the tests if unexpected calls are made
- The focus is on behaviour verification

Mock

Test Doubles: Spy

- Like a more intelligent Stub
- Keep track of how it was used
- Also helps with verification

Spy

Mockito in the sense of BDD

- TDD and BDD are totally different concepts

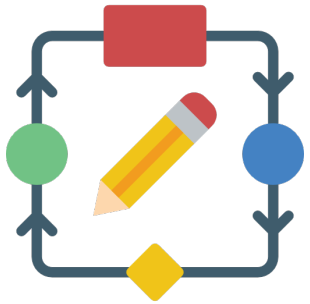
- **Arrange** vs **Given**
- **Act** vs **When**
- **Assert** vs **Then**

- BDD style introduced since 1.8.0 (MockitoBDD)

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

Seller seller = mock(Seller.class);
Shop shop = new Shop(seller);

public void shouldBuyBread() throws Exception {
  //given
  given(seller.askForBread()).willReturn(new Bread());

  //when
  Goods goods = shop.buyBread();

  //then
  assertThat(goods, containBread());
}
```

Popular annotations/methods – since 1.8

- @Mock
- @MockBean
- @Spy
- @InjectMock
- @Captor
- @Rule
- @ExtendWith(MockitoExtension.class)
- @MockitoSettings(strictness = Strictness.STRICT_STUBS)



- verify()
- verifyNoMoreInteractions()
- times()
- timeout()
- atMost()
- atMostOnce()
- atLeat()
- atLeastOnce()

Don’t’ forget!

- assert()
- assertThrows()

The DO family!

- doThrow()
- doNothing()
- doAnswer()
- doNothing()

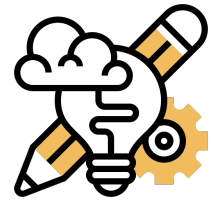
The BDD family!

- given()…then()…should()
- will()
- willAnswer()
- willReturn()
- willThrow()

Let’s do some coding

https://github.com/jerrylwang/mockito-kata

Final thoughts



- Do not mock types you don’t own
- Don’t mock value objects
- Don’t mock everything
- Run Test Coverage
- Use doReturn() in those rare occasions when you cannot use when(Object).
- when(Object) is always recommended for stubbing
- Checkout Mockito documentations for more examples

https://javadoc.io/doc/org.mockito/mockito-core/latest/org/mockito/Mockito.html
https://app.pluralsight.com/library/courses/mockito-getting-started

Questions