Testcontainers for never looking back!

Writing nice Integration Tests never was so easy!

SVQ JUG – 12/12/2019 @Seville

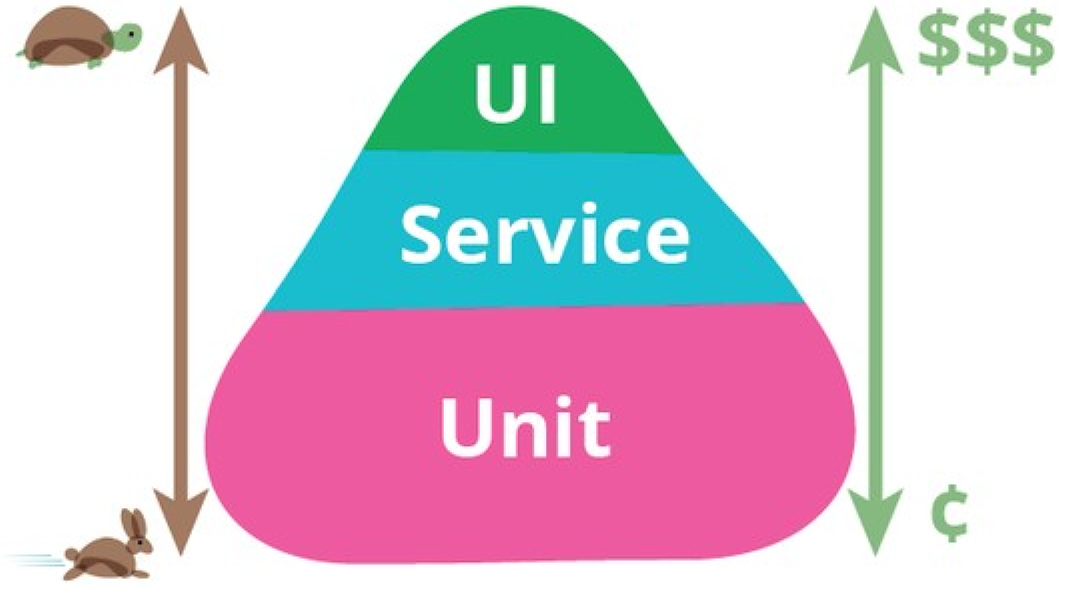
About me:

- **14 years of coding (as a professional)**
- **Now as Tech Lead and Mentor at WATA Factory**
  - But having my own personal project (KeenOn)
- **I strong believe in SOLID, TDD and Clean Code**
- **And I’m a defender of Agile against Fake Agile**

Why tests are useful?

- **We want to deliver fast**
- **But we also need to deliver without bugs**
  - So then, we need tests!

The test pyramid

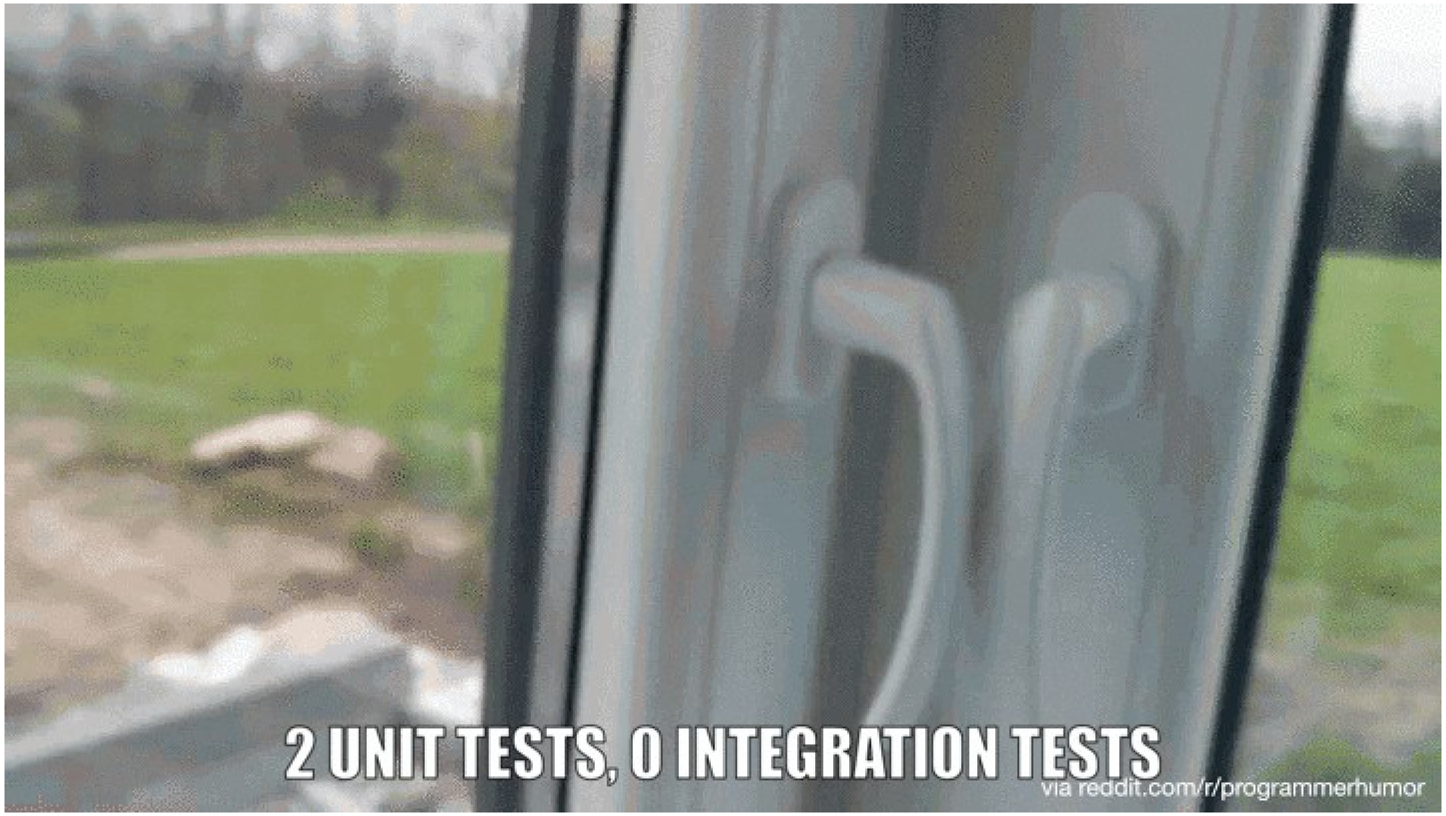


https://martinfowler.com/bliki/TestPyramid.html

UI / Service Tests

- **Aka “The selenium tests” and the “Integration Tests”**
  - The problem is they are expensive to write and also to maintenance
  - But they are needed
  - But even they're not a silver bullet to ensure no bugs ⚠

But if we only write unit tests and no integration test...

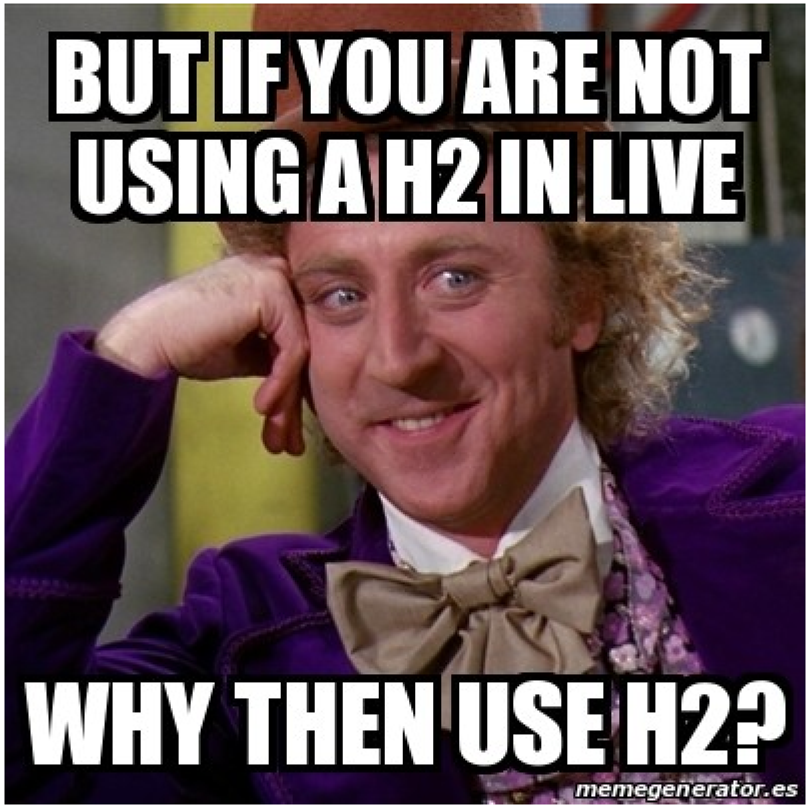

2 UNIT TESTS, 0 INTEGRATION TESTS
via reddit.com/r/programmerhumor

Some solutions to the... rescue?

Legacy or drunk solutions

- **Embedded Solutions:**
  - H2
  - SQL and NoSQL e-solutions
- **Not Embedded solutions**
  - Virtual Machines
  - “Cloning” the live environment into a CI environment

H2 Database


BUT IF YOU ARE NOT USING A H2 IN LIVE
WHY THEN USE H2?
memegenerator.es

SQL and NoSQL e-solutions

But those solutions add so much boilerplate code, and even you can't test a configuration as close as live as possible

Virtual Machines

“Cloning” the live environment into a CI environment

Consequences

- Costs money
- Costs effort
- Periodical cleanups are needed
- ...

… From the point of view of a lazy dev

- **The same old excuse “no time to write integration tests”**

Docker to the rescue!

- **Using Docker or Docker compose you can define an environment setup for your tests very similar to live**
  - Same versions of your SQL, NoSQL or whatever service do you need
  - Populating the initial data from files

But the trade off is?

- Is hard to run tests in parallel if you don't define a good "one test/one container" strategy (and this is not easy)

- If there is no "one test/one container" strategy, the data created by tests can affect each other

- You have to be aware about the tear up/down of the test

- It can leave a lot of "dead" containers at your host

And...

And here we again with the same excuse to don't write tests (!)

But now we have Testcontainers!

Testcontainers

- Only for Java (sorry)
- Reduce the boilerplate since you can define your services in an application file
- It does the tear up/down of the container used by the test by itself

Testcontainers

- **Support for SQL (MySQL, Oracle...) NoSQL(Mongo, ES...) Brokers (Kafka...) and more (Redis, RabbitMq)**

- **Even it has support for Selenium!**

... so you only need to write tests

Yay! How it works!

- **Firstly, you need to install Docker Engine at your host**
- **Add Testcontainers to your classpath**
  - Maven/Graddle
- **Define your containers**
  - application.yml
  - Programatically via Configuration Class

Example. Test Containers vs E-MySQL

Via application.yml

Programatically

E-MySQL

So no more excuses!

- **So now there are no excuses to have a good tests stack**
  - Junit
  - AssertJ
  - Rest Assured
  - Spock
  - Fluentlenium
  - and of course **Testcontainer!**

Resources:

- MySQL Testing GitHub
- https://www.testcontainers.org/
- https://www.docker.com/
- https://www.youtube.com/watch?v=Lv1evJe2MRI
- Kevin Wittek Twitter

Stay in touch

@geeksusma

Linkedin

Facebook

Questions?