Using Testcontainers and dependency breaking techniques to deal with legacy code

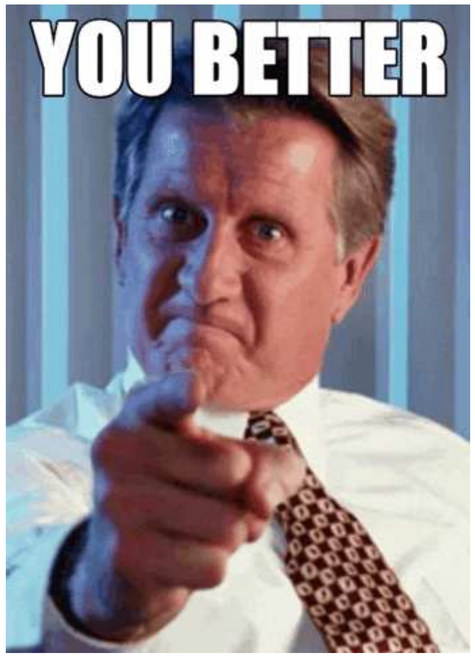
February 4, 2021

@petikoch

A BUG IN PRODUCTION!!!

- PortfolioService#printStatement is broken
-

```
company | shares | current price | current value | last operation
GameStop | 1 | $89.98 | $89.98 | SOLD 1 on 03/02/2021
```



https://gph.is/g/4A73z9q

What’s the situation?

- https://github.com/Alcor-Academy/StockPortfolio-BigBallOfMud-CSS-01
- My findings:
  - It has 0 (ZERO!!!) automated tests!
  - Many (MANY!!!) legacy code smells (http://legacycodesmells.com/)
  - At least 1 serious bug in production

http://gph.is/2neJxsZ

Let’s relax.

Remember the ALCOR.academy training? Code Renovation Modul?

http://gph.is/1PRG4Xg

One guy mentioned „Testcontainers“

… during the ALCOR.academy training

- https://www.testcontainers.org/

testcontainers/testcontainers-java is licensed under the
MIT License

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- ✓ Commercial use
- ✓ Modification
- ✓ Distribution
- ✓ Private use

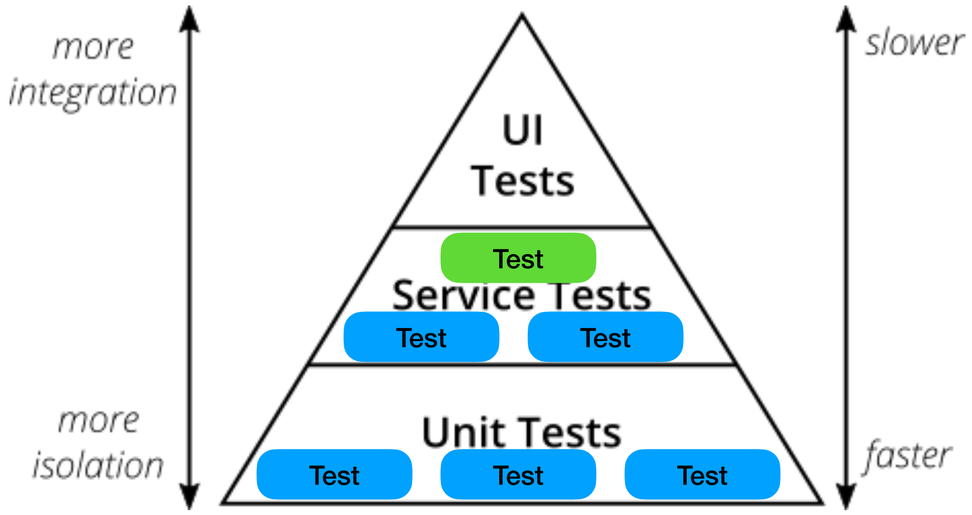
Limitations

- ✗ Liability
- ✗ Warranty

Our next experiment!

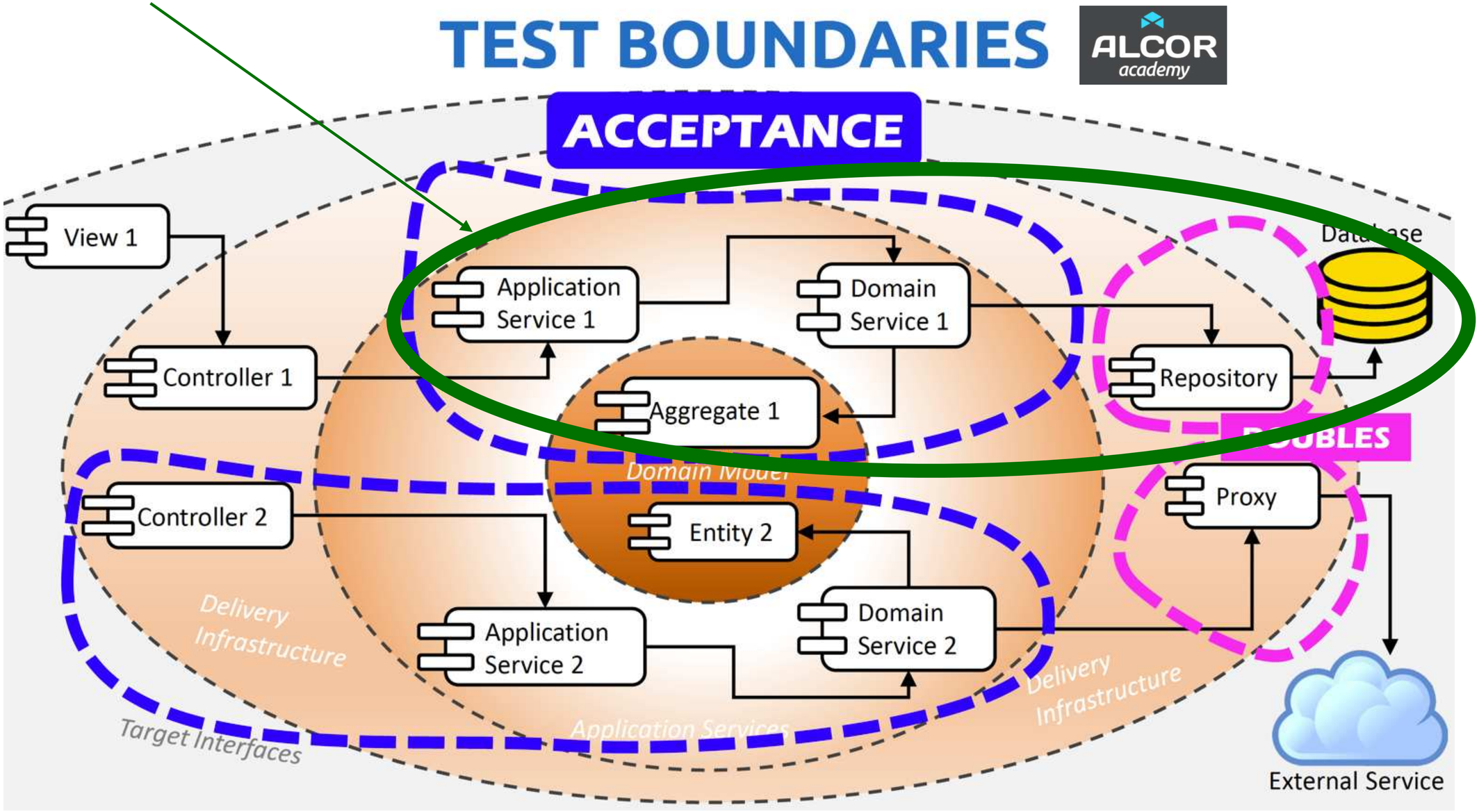
We'll give Testcontainers a try today.

- Step 1 (today)
  - Get characterization tests in place by using Testcontainers and do only the **absolute minimum of refactoring**
  - We'll have a first safety net, then!
- Step 2 (tomorrow or so?)
  - Do more refactoring as needed (break dependencies…)
  - Add unit tests and acceptance tests without MongoDB integration



https://martinfowler.com/articles/practical-test-pyramid.html

Test Scope


TEST BOUNDARIES
ALCOR academy
ACCEPTANCE
View 1
Controller 1
Controller 2
Application Service 1
Application Service 2
Domain Service 1
Domain Service 2
Aggregate 1
Entity 2
Repository
Proxy
Database
DOUBLES
Domain Model
Delivery Infrastructure
Delivery Infrastructure
Target Interfaces
Application Services
External Service

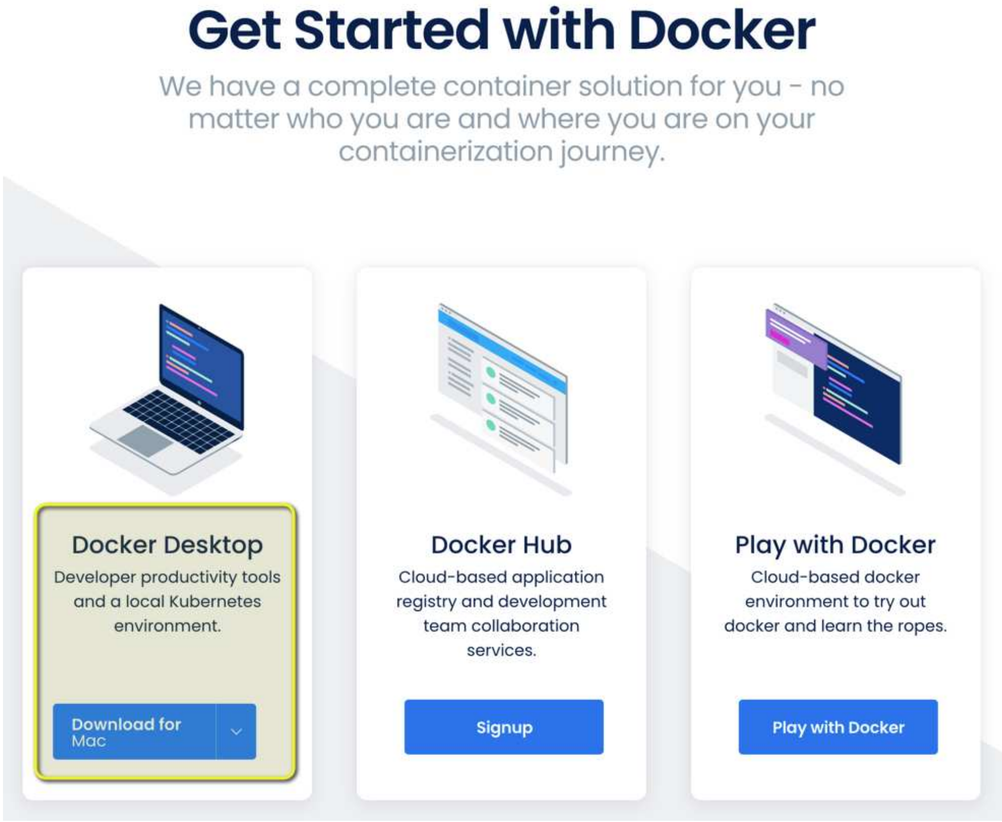
Goals for step 1

- ☑ Test on API level with good coverage
- ☑ with full MongoDB integration
- ☑ decent fast tests (<= 5 seconds per test)
- ☑ minimal changes to the existing code
- ☑ **finish in less than 10 minutes**

We need a Docker daemon instance

… on our local dev machine

- See https://www.testcontainers.org/#prerequisites
- For our local dev machine, easy!
- https://www.docker.com/get-started
- All major OS supported
  - Mac
  - Windows
  - Linux



We need a Docker daemon instance

… also on the CI/CD-Servers

- Install Docker on the CI/CD hosts itself
- Or provide a remote Docker daemon instance for them
  (for e.g. cloud native builds on Kubernetes)

So, let’s start!

https://gph.is/g/ZWdK71X

Let’s add Testcontainers to the Maven build

- pom.xml

```
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.7.0</version>
        </dependency>
        <dependency>
            <groupId>org.testcontainers</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>1.15.1</version>
        </dependency>
        <dependency>
            <groupId>org.testcontainers</groupId>
            <artifactId>mongodb</artifactId>
            <version>1.15.1</version>
        </dependency>
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-core</artifactId>
            <version>3.7.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Let’s try Testcontainers with MongoDB

```java
import org.junit.jupiter.api.Test;
import org.testcontainers.containers.MongoDBContainer;
import org.testcontainers.junit.jupiter.Container;
import org.testcontainers.junit.jupiter.Testcontainers;
import org.testcontainers.utility.DockerImageName;

import static org.junit.jupiter.api.Assertions.*;

@Testcontainers
class PortfolioServiceShould {

    @Container
    private final MongoDBContainer mongoDBContainer = new MongoDBContainer("mongo:4.4.3");

    @Test
    void demo_testcontainers() {
        assertTrue(mongoDBContainer.isRunning());
        int mongoDbPort = mongoDBContainer.getFirstMappedPort();
        assertTrue(mongoDbPort >= 1 && mongoDbPort <= 65535);
    }

}
```

What we learned: Break the dependency!

… to the hardcoded MongoDB URL

Before

```
public class PortfolioService {
    private static final MongoClient mclSingle = MongoClients.create(Globals.DBConf[1]);
    private final int userId;
    private final MongoDatabase db;
    private static final String loggerDbName = Globals.DBConf[0];
    private static final Logger mongoLogger = Logger.getLogger(loggerDbName);

    public PortfolioService(int userId) {
        this.userId = userId;
        mongoLogger.setLevel(Level.SEVERE);
        db = mclSingle.getDatabase(Globals.DBConf[2]);
    }
```

After

```
public class PortfolioService {
    private final int userId;
    private final MongoDatabase db;
    private static final String loggerDbName = Globals.DBConf[0];
    private static final Logger mongoLogger = Logger.getLogger(loggerDbName);

    public PortfolioService(int userId) {
        this(userId, Globals.DBConf[1]);
    }

    PortfolioService(int userId, String mongoDbUrl) {
        this.userId = userId;
        mongoLogger.setLevel(Level.SEVERE);
        MongoClient mclSingle = MongoClients.create(mongoDbUrl);
        db = mclSingle.getDatabase(Globals.DBConf[2]);
    }
```

Result

```
@Testcontainers
class PortfolioServiceShould {

    @Container
    private final MongoDBContainer mongoDBContainer = new MongoDBContainer("mongo:4.4.3");

    @Test
    void printStatement_prints_BUY_for_1_buy_transaction() {
        PortfolioService portfolioService =
                new PortfolioService(1, "mongodb://localhost:" + mongoDBContainer.getFirstMappedPort());
        portfolioService.buyShares(1, "GameStop");

        String statement = portfolioService.printStatement();

        String scrubbedStatement = statement
                .replaceAll("\\d+\\.\\d+", "__.__")
                .replaceAll("\\d{2}/\\d{2}/\\d{4}", "__/__/____");
        assertEquals(
                "company | shares | current price | current value | last operation\n" +
                "GameStop | 1 | $__.__ | $__.__ | SOLD 1 on __/__/____\n",
                scrubbedStatement
        );
    }

}
```

Conclusion

- [x] It’s easy and quick to setup characterization tests with that scope using Testcontainers
- [x] Wo got some initial coverage —> safety net from the beginning
- [x] Good first step
- [ ] Step 2 likely to follow
  - [ ] Fast acceptance tests without MongoDB integration
  - [ ] Unit tests
  - [ ] Break more dependencies (Time, Price, …)
    to get more coverage

http://gph.is/1FA0WT1

Questions?

- References
  - https://www.testcontainers.org/
  - https://alcor.academy/code-renovation
  - https://giphy.com

@petikoch

http://gph.is/2gWzAAh