Next Gen Testing Tools for Java

Daniel Hinojosa

https://github.com/dhinojosa/next-gen-testing-tools-java

Introduction

Daniel Hinojosa
Programmer, Consultant, Trainer
evolutionnext.com

Testing in Scala (Book)
Beginning Scala Programming (Video)
Scala Beyond the Basics (Video)

Contact:
dhinojosa@evolutionnext.com
@dhinojosa@mastodon.social

Property Testing with JQwik

What is a Property?

What is a property, and how is this different than a unit-test?

- This doesn’t make a direct assignment of value
- I want to make sure a certain rule about my software holds
- These rules are called properties
- A property is different from an assertion in that we are testing a principle or characteristic of a function ***against itself***

Questions: What Properties can you come up with?

Challenge: Reversing a String

Challenge: Concatenating Two Lists

Challenge: Building a Custom Map

Challenge: Absolute Value of a Number

Fizz Buzz Test

The "Fizz-Buzz test" is an interview question designed to help filter out the 99.5% of programming job candidates who can't seem to program their way out of a wet paper bag. The text of the programming assignment is as follows:

> *"Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz"."*

Don't believe it,
it's more like 60%

Challenge: How would do the property for FizzBuzz?

Challenge: Fizz Buzz

- All divisible by 3 values should start with "fizz"
- All divisible by 5 values should end with "buzz"

What property test will not give you

- Not a replacement for Test Driven Development
- It is not a randomizer of input, it has a specific need to find outliers
- `0`, `1`, `-1`, `Integer.MAX_VALUE`, `Integer.MIN_VALUE`

JQwik

- https://jqwik.net/
- Bring Property-Based Testing (PBT) to the JVM
- Supports JUnit5
- Container Reporting
- Supports Arbitrary Data Creation
- Supports Result Shrinking

Demo: JQwik

Mutation Testing with PITest

PiTest and Mutation Testing

- https://pitest.org/
- PITest is made for JUnit 4 but has extensions to work with JUnit5.
  - A plugin is available at https://github.com/pitest/pitest-junit5-plugin
- Faults (or mutations) are automatically seeded into your code as your tests are run.
- If your tests fail, the mutation is killed; if your tests pass, the mutation lives.
- Run it frequently against only the code that has been changed.

Hello, I am here to destroy your code.

```
122                     // Verify for a ".." component at next iter
123 3                   if ((newcomponents.get(i)).length() > 0 &
124                     {
125                         newcomponents.remove(i);
126                         newcomponents.remove(i);
127 1                       i = i - 2;
128 1                       if (i < -1)
129                         {
130                             i = -1;
131                         }
132                     }
133                 }
```

Maven Install

```
<plugin>
    <groupId>org.pitest</groupId>
    <artifactId>pitest-maven</artifactId>
    <version>LATEST</version>
</plugin>
```

- You certainly have many options other than the base install
  - Targeting specific classes
  - Targeting specific goals
  - Establishing reports

```
mvn org.pitest:pitest-
maven:mutationCoverage
```

Demo: PITest

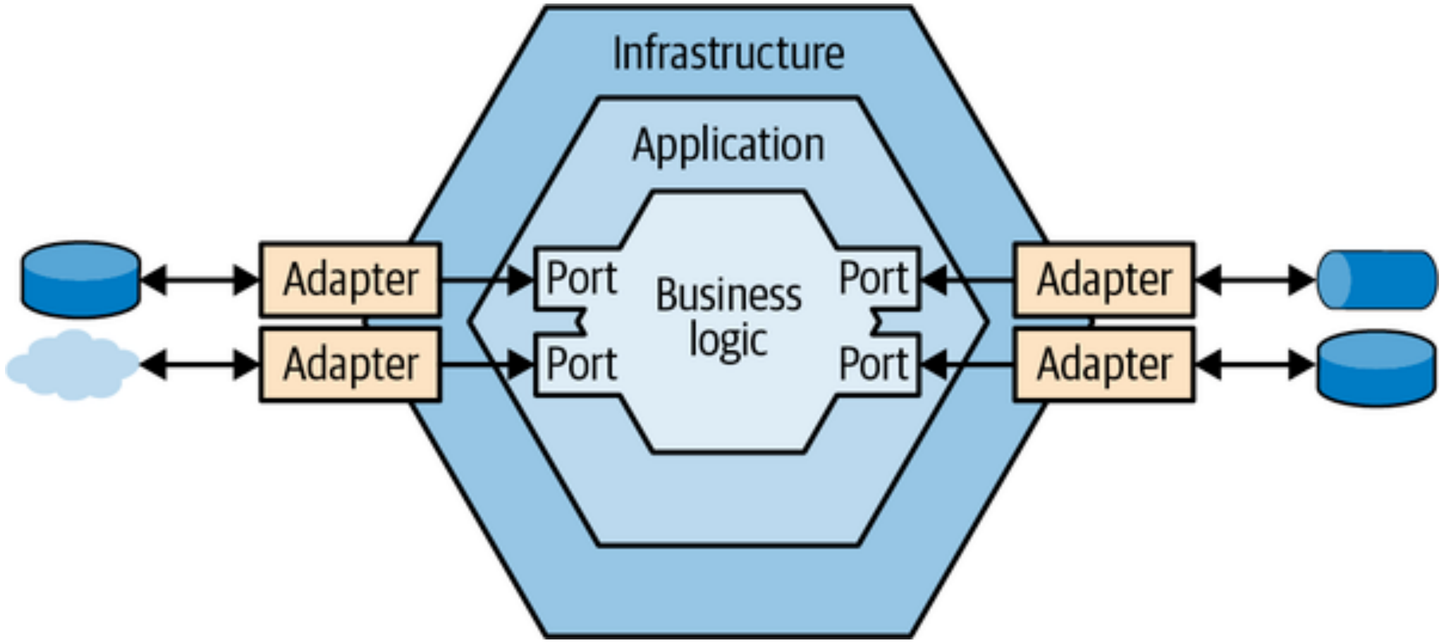
Database Testing with Test Containers

Hexagonal Architecture

- Ports and Adapters Architecture
- Decouple the system's business logic from its infrastructural components
- The infrastructure layer implements "adapters", which are concrete implementations

- Originally conceptualized by Alistair Cockburn
- Compliments Domain Driven Design
- Separation of Concerns. Domain is encapsulated and interacts through the ports and adapters

Hexagonal Architecture



Learning Domain Driven Design - Vlad Khononov

Interfaces

- Data Access Objects are defined by interfaces from the domain
- Domain Determines what is required, and the DAO should adhere to the contract
- The DAO can have many implementations

What is it?

- Testcontainers is a Java library that uses containers to perform testing against common systems
  - https://www.testcontainers.org/
  - Supports JUnit, Spock testing
  - Lightweight
  - Ephemeral
  - Docker-based
  - Better than doing DBUnit

Why does it make it easier?

- Use a containerized instance of a MySQL, PostgreSQL or Oracle database to test your data access layer code for complete compatibility, but without requiring complex setup
- Running your application in a short-lived test mode with dependencies, such as databases, message queues or web servers.
- Use containerized web browsers, compatible with Selenium, for conducting automated UI tests. Each test can get a fresh instance of the browser, with no browser state
- Much more! Check out the various contributed modules or create your own custom container classes using `GenericContainer` as a base.

London vs. Chicago Style TDD

London vs. Chicago Style TDD

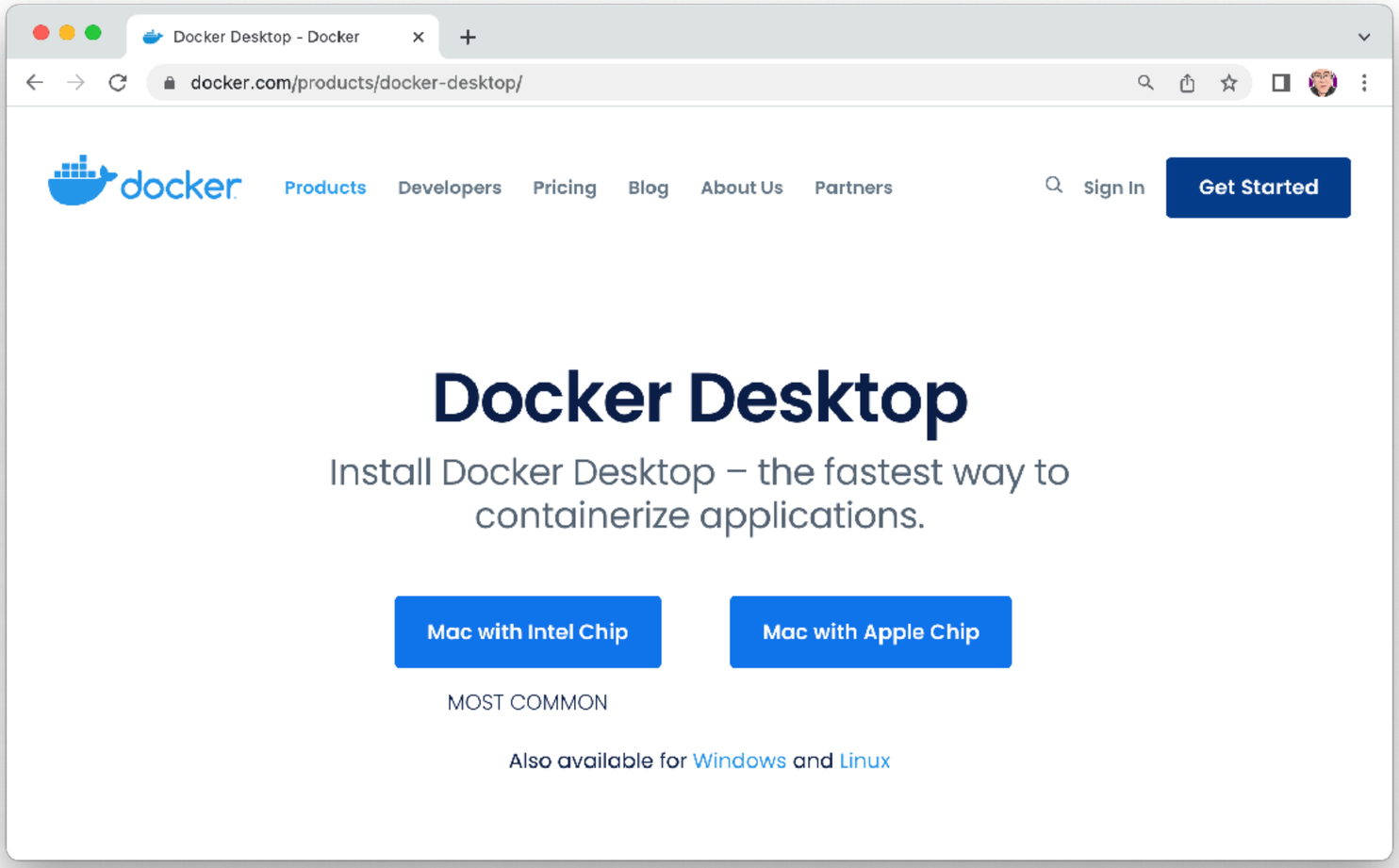
- Chicago [Classic] TDD - State-based behavior testing and triangulation
- London [Mockist] TDD - Interaction testing, mocking and end-to-end TDD

“Don’t [unit] test what you don’t own”

Growing Object-Oriented Software Guided By Tests - Nat Pryce

Installation

https://www.docker.com/products/docker-desktop/



Maven BOM

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.testcontainers</groupId>
            <artifactId>testcontainers-bom</artifactId>
            <version>1.17.5</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

Maven Dependency

```
<dependency>
    <groupId>org.testcontainers</groupId>
    <artifactId>testcontainers</artifactId>
    <version>1.17.5</version>
    <scope>test</scope>
</dependency>
```

This is the minimal requirement needed for JUnit 4

Gradle Dependency

```
testImplementation "org.testcontainers:testcontainers:1.17.5"
```

This is the minimal requirement needed for JUnit 4

Maven Dependency with JUnit 5

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.8.1</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.testcontainers</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>1.17.5</version>
    <scope>test</scope>
</dependency>
```

Gradle Dependency with JUnit 5

```
testImplementation "org.junit.jupiter:junit-jupiter:5.8.1"
testImplementation "org.testcontainers:testcontainers:1.17.5"
testImplementation "org.testcontainers:junit-jupiter:1.17.5"
```

Spock Dependency

```
testImplementation "org.testcontainers:spock:1.17.5"
```

Demo: Test Containers

MockServer

Mock-Server

- https://www.mock-server.com/
- Mock Server used for testing that can:
  - Return mocked or canned answers as if this is a web server; well, it is.
  - It acts as a proxy to a genuine website to record any payload so you can test against it.
  - A mixture of both mocked or proxied behavior
- Supports Java and JavaScript

A Note about Parsing

- Parsing JSON, XML, et al. *should be done as a unit-test*
- How you convert your payload into a Data Transfer object should have nothing to do with the network or any protocol.
- Keep in mind that Java has one of the most significant productivity gains since JDK 15 regarding payload parsing and testing...

Demo: MockServer

Restful Testing with Rest Assured

Rest-Assured

- https://rest-assured.io/
- API Testing Native to Java
- “But I love Postman.”
- This is a part of your code, but likely in its own project
- Subsequently, it will be a part of your CI/CD although shift right, after staging and after production

What levels of API Testing are there?

- **Baseline API Testing -** Does it meet the essential criteria
- **API Contract Testing -** Does the API meet contract criteria
- **API Regression Testing -** We found the bug, and we created the test to verify the failure; let’s keep it so we can avoid future failures
- **API Reliability Testing -** Are the results always consistent
- **API Monitoring Testing -** Are the APIs working as expected

Maven Install

```
<dependency>
	<groupId>io.rest-assured</groupId>
	<artifactId>rest-assured</artifactId>
	<version>5.2.0</version>
	<scope>test</scope>
</dependency>
```

Gradle Install

```
testImplementation 'io.rest-assured:rest-assured:5.2.0'
```

JSON Path Maven

```
<dependency>
	<groupId>io.rest-assured</groupId>
	<artifactId>json-path</artifactId>
	<version>5.2.0</version>
	<scope>test</scope>
</dependency>
```

- Used for JSON Parsing
- Uses Groovy’s GPath Syntax

JSON Path Gradle

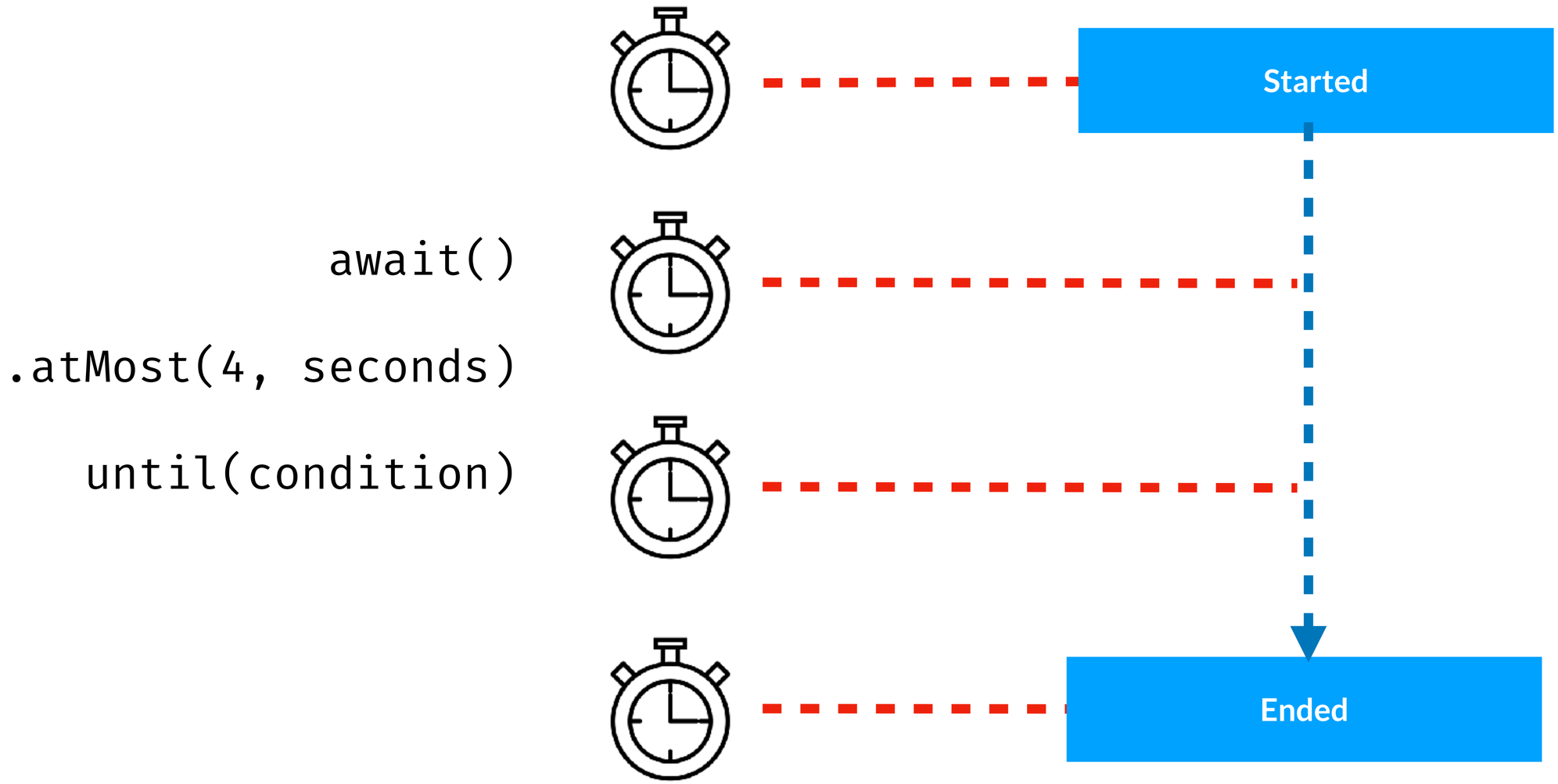
```
testImplementation 'io.rest-assured:json-path:5.2.0'
```

Demo: Test Rest-Assured

Asynchronous Testing with Awaitility

Awaitility

- https://github.com/awaitility
- Domain Specific Language Allows you to express expectations
  - The perfect tool for handling threads, timeouts, and concurrency issues
- Performs polling until a particular *effect is completed*


Started
await()
.atMost(4, seconds)
until(condition)
Ended

What is the condition?

- For every beginning, there is an end.
- For every asynchronous process that begins, there must be an end.
- Let me ask you: What are some examples where you can observe the effect of an asynchronous process?
- We also can’t wait forever, so we will need to cap the time or else we fail.

Maven Install

```
<dependency>
      <groupId>org.awaitility</groupId>
      <artifactId>awaitility</artifactId>
      <version>4.2.0</version>
      <scope>test</scope>
</dependency>
```

Gradle Install

```
testImplementation 'org.awaitility:awaitility:4.2.0'
```

Various DSLs Are Available

Demo: Awaitility

Thank You

- Email: dhinojosa@evolutionnext.com
- Github: https://www.github.com/dhinojosa
- Mastodon: @dhinojosa@mastodon.social
- Linked In: http://www.linkedin.com/in/dhevolutionnext