Introducción a los Tests Unitarios en Java

con JUnit 5

Luis Viñé

Desarrollo AlphaEscapex

Mayo 2023

INDICE

Unit Tests en JAVA

En JAVA la librería JUnit lleva muchos años funcionando.
La última versión es JUnit 5.
Dependencias Maven integradas en Spring para Tests

Tests Unitarios

Todo buen desarrollador de software debe saber cómo escribir tests unitarios... ¡y escribirlos!

- Tests al más bajo nivel.
- Prueban una única unidad de software.
- En Java, normalmente un método de una clase.

Ejemplo

Método:

```
1  /**
2  * Convierte grados Fahrenheit a Celsius
3  */
4  double asCelsius(double temperatureFahrenheit);
```

Test Unitario:

```
1  @Test
2  void should_ReturnCorrectTemperatureAsCelsius() {
3     assertEquals(asCelsius(41.0), 5.0);
4  }
```

¿Son necesarios?

Los tests unitarios son NECESARIOS en toda buena pieza de software.

- Mayor confianza durante el desarrollo
- Código más limpio y reutilizable
- Menor coste debido a errores

¿Quién escribe test unitarios?

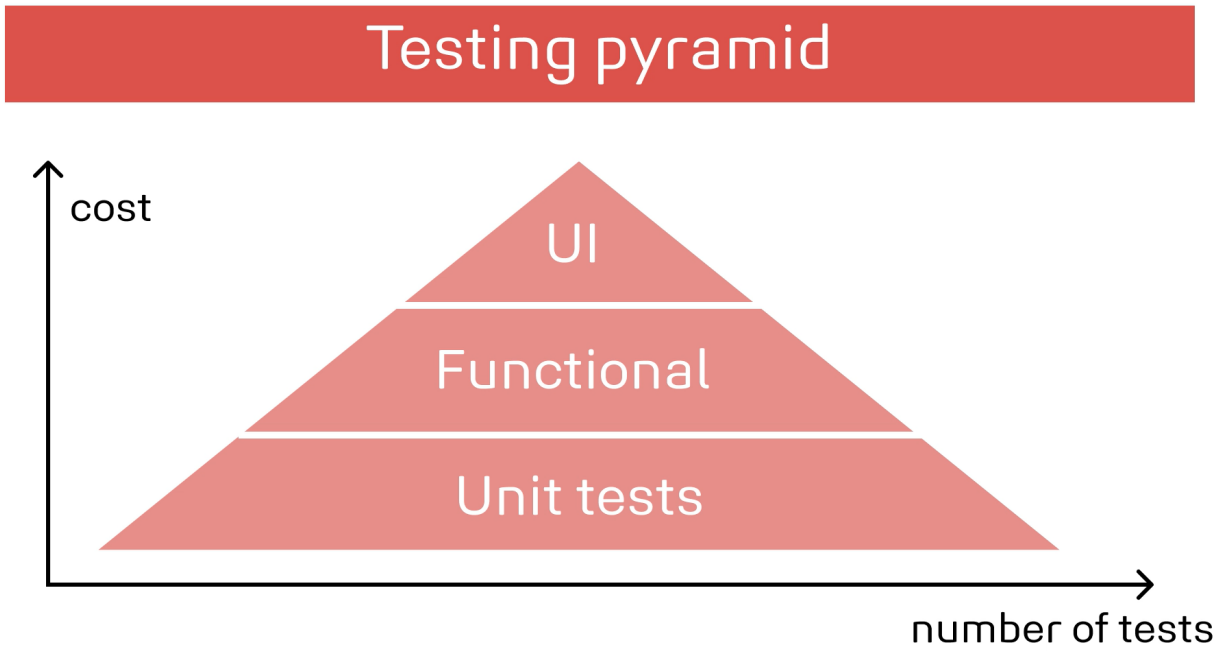


Figura:

Típicamente escritos por Desarrolladores, no por Testers.

Frameworks

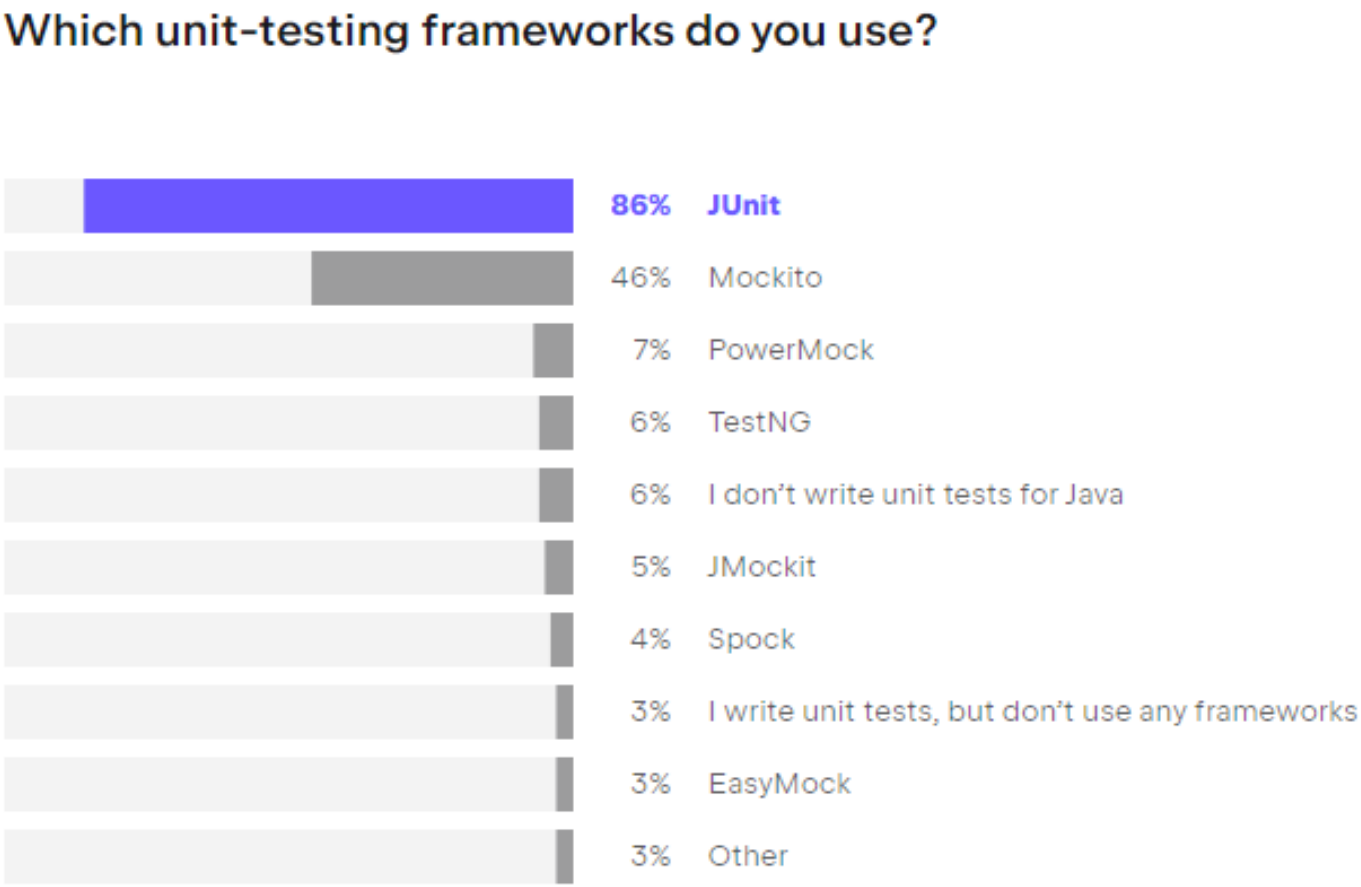
- JUnit - framework estándar para Java
- Combinado a menudo con Mockito
- JUnit 4 (2006)
- JUnit 5 "Jupiter" (2017)

Frameworks

- JUnit - framework estándar para Java
- Combinado a menudo con Mockito
- JUnit 4 (2006)
- JUnit 5 "Jupiter" (2017)

Frameworks

- JUnit - framework estándar para Java
- Combinado a menudo con Mockito
- JUnit 4 (2006)
- JUnit 5 "Jupiter" (2017)



Fuente: `https://www.jetbrains.com/lp/devecosystem-2022/java/`

Mejores prácticas

- El nombre debe describir lo que hace el test
- Convención: *should_x_When_y*
  - *should_ReturnTrue_When_DietRecommended()*
- El cuerpo del test debe ser informativo
- Convención:
  - given - when - then
  - Arrange - Act - Assert
- Comprobar los casos normales
- ...y también los errores

INDICE

Anotaciones

Anotación	Descripción
@Test	Marca un método como Test
@ParameterizedTest	Marca un método como Test Parametrizado
@RepeatedTest	Repite un test N veces
@TestFactory	Método que devuelve tests, para test dinámicos
@TestInstance	Sirve para configurar el ciclo de vida de una instan

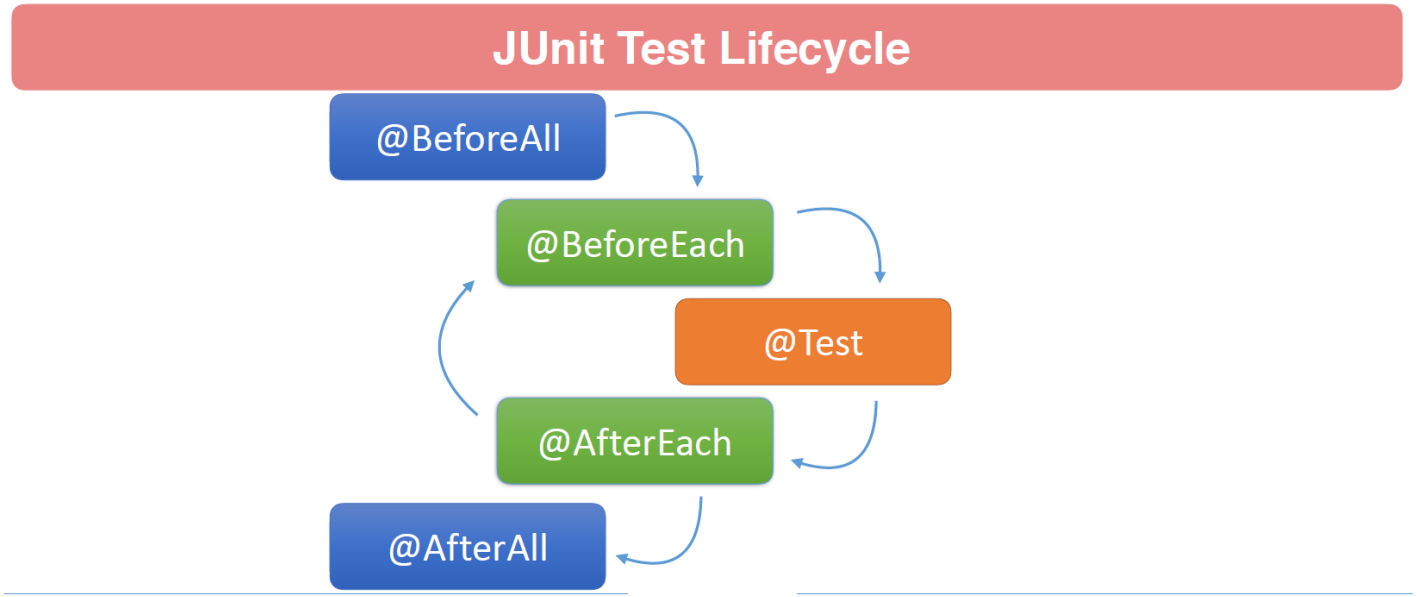
Anotaciones

Anotación	Descripción
@TestTemplate	Crea una plantilla para usarla en múltiples test
@DisplayName	Para mostrar un nombre inteligible para un test
@BeforeEach	Método que se ejecuta antes de cada test
@AfterEach	Método que se ejecuta después de cada test
@BeforeAll	Método estático que se lanza antes que todos los tests

Anotaciones

Anotación	Descripción
@AfterAll	Método estático que se lanza después de todos los tests
@Nested	Crea una clase de test anidada
@Tag	Declara etiquetas para filtrar los tests
@Disabled	Deshabilita un test o la clase entera
@ExtendWith	Permite registrar extensiones

Ciclo de vida



INDICE

Maven: Dependencias

- junit-jupiter-engine: el corazón de la máquina
- junit-jupiter-api: proporciona el API (dependencia transitiva)
- junit-jupiter-params: si vamos a usar tests parametrizados

Maven: Dependencias

Referencia

https://maven.apache.org/surefire/maven-surefire-plugin/examples/junit-platform.html

https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-params

Maven: Plugins

Dos plugins importantes para Tests:

- Maven Surefire Plugin - Tests Unitarios
- Maven FailSafe Plugin - Tests de Integración

INDICE

S.O.L.I.D.

- Single Responsibility
- Open-Closed
- Liskov Substitution
- Interface Segregation
- Dependency Inversion

Referencia

`https://www.baeldung.com/solid-principles`

Design Patterns

- Recetario"de soluciones para problemas comunes
- Concepto de arquitectura portado al desarrollo de software
- Usar los medicamentos con moderación

Referencia

`https://www.baeldung.com/design-patterns-series`

"Gang of Four" (Gamma y otros) - Design Patterns: Elements of Reusable Object-Oriented Software, 1994

Refactoring

El software no es estático, sino que evoluciona en el tiempo según las necesidades. Los tests garantizan que los cambios no rompan la funcionalidad existente.

Referencia

`https://www.baeldung.com/cs/refactoring`

TDD, BDD

Referencia

https://www.paradigmadigital.com/dev/tdd-como-metodologia-de-diseno-de-software/

https://www.itdo.com/blog/que-es-bdd-behavior-driven-development/

Agile

Agile integra las prácticas anteriores y más cosas...

- Equipo - Cliente
- Springs
- Metodología (Kanban, SCRUM,...)

Referencia

`https://www.agilealliance.org/agile101/`