Testing Efectivo en Java

Círculo Siete Capacitación

Clase 5 de 12

14 Agosto 2025

Introducción a Testcontainers para Java

- Testcontainers es una biblioteca de Java que permite levantar contenedores de Linux (Docker) de forma programática durante la ejecución de pruebas automatizadas.
- Su objetivo principal es facilitar tests de integración y end-to-end al proporcionar entornos reales (bases de datos, colas, servicios externos, etc.) de manera reproducible y aislada.

Ventajas clave

- **Entornos realistas**: No dependes de simulaciones o configuraciones locales inciertas; pruebas contra servicios reales.
- **Reproducibilidad**: Cada ejecución levanta un entorno limpio y controlado.
- **Portabilidad**: Funciona en cualquier máquina con un container runtime, ya sea local o en CI.
- **Integración transparente**: Compatible con JUnit 4 y JUnit 5.

Casos de uso comunes

- Bases de datos para pruebas
  - Ejemplo: PostgreSQL, MySQL, MongoDB, Redis.
- Mensajería y streaming
  - Ejemplo: Kafka, RabbitMQ.
- Sistemas externos simulados
  - Ejemplo: LocalStack (servicios AWS simulados).
- Infraestructura personalizada
  - Levantar cualquier imagen OCI que tu aplicación necesite para probar.

¿Cómo funciona?

- Testcontainers utiliza el cliente oficial de Docker para levantar contenedores definidos como instancias Java.
- En cada prueba:
  - Descarga (si es necesario) la imagen OCI.
  - Inicia el contenedor antes de la prueba.
  - Expone sus puertos y variables necesarias.
  - Lo detiene automáticamente al finalizar.

Agregar dependencia en Maven

```xml
<dependency>
    <groupId>org.testcontainers</groupId>
    <artifactId>postgresql</artifactId>
    <version>1.20.1</version>
    <scope>test</scope>
</dependency>
```

Test de integración

```java
import org.junit.jupiter.api.Test;
import org.testcontainers.containers.PostgreSQLContainer;
import org.testcontainers.junit.jupiter.*;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;

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

@Testcontainers
public class PostgresIntegrationTest {

    @Container
    static PostgreSQLContainer<?> postgres = new PostgreSQLContainer<>("postgres:16-alpine")
            .withDatabaseName("testdb")
            .withUsername("testuser")
            .withPassword("testpass");

    @Test
    void testConnection() throws Exception {
        try (Connection conn = DriverManager.getConnection(
                postgres.getJdbcUrl(),
                postgres.getUsername(),
                postgres.getPassword())) {
            ResultSet rs = conn.createStatement().executeQuery("SELECT 1");
            rs.next();
            assertEquals(1, rs.getInt(1));
        }
    }
}
```

Puntos clave

- ***@Testcontainers*** y ***@Container*** gestionan automáticamente el ciclo de vida del contenedor.
- Se pueden usar contenedores estáticos (iniciados una vez para toda la clase) o de instancia (uno por test).
- El código es idéntico a cómo conectarías a una base de datos real.

Integración en CI/CD

- Testcontainers funciona en la mayoría de plataformas CI (GitHub Actions, GitLab CI, Jenkins).
- Asegúrate de que el entorno tenga Docker disponible o usa Docker-in-Docker.
- Útil para tests de contrato, tests de integración cross-service y validación previa a despliegue.

Notas finales

- Testcontainers es una herramienta clave para realizar pruebas de integración realistas en Java, eliminando la dependencia de entornos manuales y facilitando la reproducibilidad de las pruebas.
- Permite que tu código de pruebas hable con servicios reales, lo que aumenta la confianza en que funcionará en producción.

Laboratorio

- Revisión de tarea de la clase 3
- Pruebas de integración
  - labs