Testing con Microservicios

JoeDayz | 2025

Quarkus-junit5

Notas

- Soporte a CDI
- Testing HTTP simplificado
- Mocking simplificado
- Servicios contenerizados
- Requisito: implementar el Surefire maven plugin, el log manager.

```java
import javax.inject.Inject;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;
import io.quarkus.test.junit.QuarkusTest;

@QuarkusTest (1)
public class MyEngineTest {

    @Inject
    Engine engine; (2)

    @Test (3)
    public void testEngine() {
        // Given
        engine.stop();

        // When
        engine.start();

        // Then
        Assertions.assertTrue(engine.isRunning()); (4)
    }
}
```

Nota: tambien puede usar Hamcrest, AssertJ

DEV SERVICES

```
quarkus.datasource.db-kind=postgresql (1)
%test.quarkus.datasource.db-kind=h2 (2)
```

Recursos

https://junit.org/junit5/

https://github.com/rest-assured/rest-assured

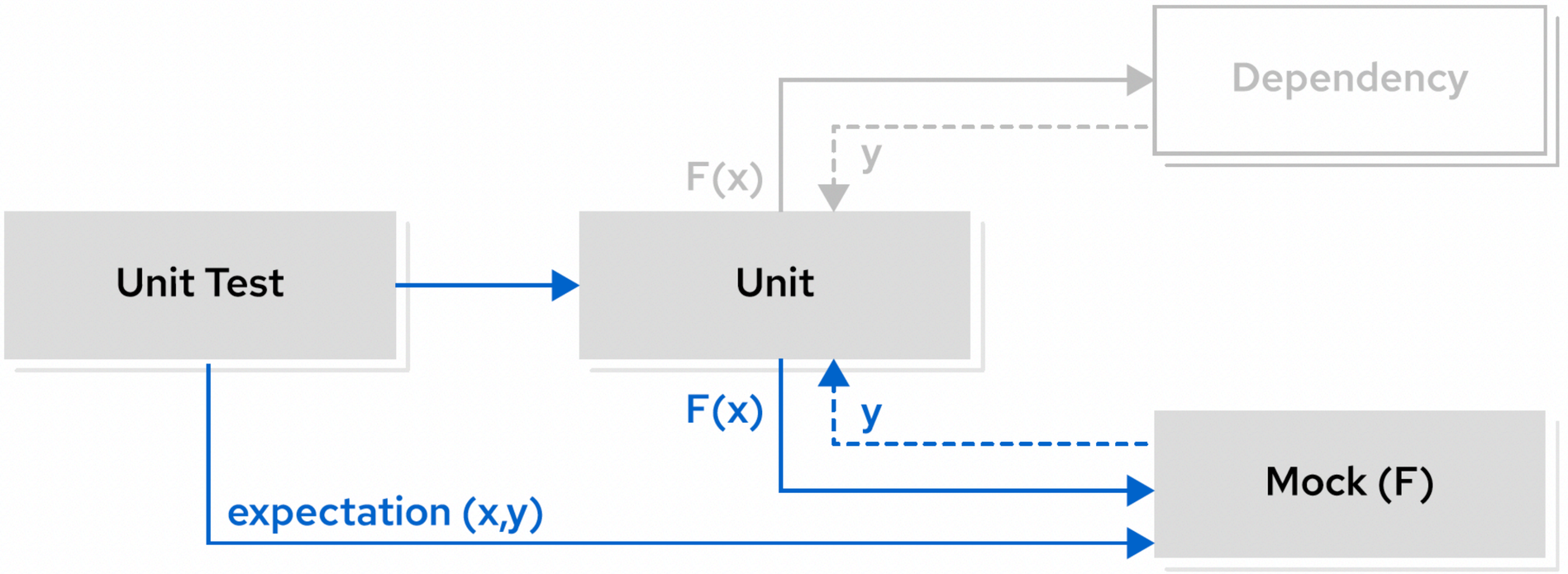

Figure 3.1: Mock type example

```
import io.quarkus.test.Mock;

import javax.enterprise.context.ApplicationScoped;

@Mock (1)
@ApplicationScoped (2)
public class OrderServiceMock extends OrderService {
    @Override
    public Order getOrder(UUID uuid) {
        return new Order(uuid, "Customer Name", 123);
    }
}
```

```
@QuarkusTest
public class OrderServiceTest {

    @Inject (1)
    OrderService orderService;

    @Test
    public void testExample() {
        UUID uuid = ...code omitted...

        Order order = orderService.getOrder(uuid); (2)

        boolean isFraud = orderService.isFraud(order); (3)

        ...code omitted...
    }
}
```

```
@QuarkusTest
public class OrderServiceTest {
    @Test
    public void testExample() {
        UUID uuid = ...code omitted...

        OrderService orderService = Mockito.mock(OrderService.class); (1)

        Mockito
            .when(orderService.getOrder(uuid)) (2)
            .thenReturn(new Order(uuid, "Customer Name", 123)); (3)

        Mockito
            .when(orderService.getOrder(null)) (4)
            .thenThrow(new NullPointerException()); (5)

        Order order = orderService.getOrder(uuid); (6)

        ...code omitted...

        Order order = orderService.getOrder(null); (7)

        ...code omitted...
    }
}
```

```
@QuarkusTest
public class OrderServiceTest {
    @Test
    public void testExample() {
        UUID uuid = ...code omitted...

        OrderService orderService = Mockito.mock(OrderService.class);

        Mockito
            .when(orderService.getOrder(Mockito.any())) (1)
            .thenReturn(new Order(uuid, "Customer Name", 123));

        Order order = orderService.getOrder(uuid); (2)
        Order order = orderService.getOrder(null); (3)

        ...code omitted...
    }
}
```

```
@QuarkusTest
public class SomeServiceTest {

    @InjectMock (1)
    OrderService orderService;

    @BeforeEach
    public void setup() {
        Mockito
            .when(orderService.getOrder(null))
            .thenThrow(new NullPointerException()); (2)
    }

    @Test
    public void testExample() {
        UUID uuid = ...code omitted...

        Mockito
            .when(orderService.getOrder(Mockito.any()))
            .thenReturn(new Order(uuid, "Customer Name", 123)); (3)

        Data data = orderService.getOrder(uuid);

        ...output omitted...
  }
}
```

@InjectMock

Testing Indirect Interactions

```
@QuarkusTest
public class SpyTest {
    @InjectMock
    OrderService orderService;

    @Test
    public void testExample() {
        given()
        .when()
            .get("/orders")
        .then()
            .statusCode(200)
            .body("$.size()", is(0));

        Mockito.verify(orderService, Mockito.times(1)).list(); (1)
        Mockito.verify(expenseService).list(); (2)
        Mockito.verifyNoMoreInteractions(orderService, expenseService); (3)
    }
}
```

Mocking de Panache Entities

```java
@QuarkusTest
public class SomeResourceTest {

    @BeforeAll
    public static void setup() {
        PanacheMock.mock(Order.class); (1)
    }

    @Test
    public void testExample() {
        Mockito
            .when(Order.listAll()) (2)
            .thenReturn(Collections.emptyList()); (3)

        given() (4)
        .when()
            .get("/orders")
        .then()
            .statusCode(200)
            .body("$.size()", is(0));
    }
}
```

```xml
<dependency>
  <groupId>io.quarkus</groupId>
  <artifactId>quarkus-panache-mock</artifactId>
  <scope>test</scope>
</dependency>
```

Mocking de REST Clients

```
@QuarkusTest
public class SomeResourceTest {

    @InjectMock
    @RestClient
    FraudScoreService scoringService; (1)

    @Test
    public void testExample() {
        UUID uuid = ...code omitted...

        Mockito
            .when(scoringService.getById(Mockito.anyString())) (2)
            .thenReturn(new FraudScore(uuid, 9999)); (3)

        given() (4)
        .when()
            .get("/orders/score/an-order-uuid")
        .then()
            .statusCode(200)
            .body("$.size()", is(1));
    }
}
```

Spies

```
@QuarkusTest
public class SomeResourceTest {

    @InjectSpy
    OrderService orderService; (1)

    @Test
    public void testExample() {
        given() (2)
        .when()
            .get("/orders")
        .then()
            .statusCode(200)
            .body("$.size()", is(0));

        Mockito.verify(orderService, Mockito.times(1)).list(); (3)
    }
}
```

Recursos

https://jakarta.ee/specifications/cdi/2.0/cdi-spec-2.0.html

https://site.mockito.org/

Dev Services

```
quarkus.devservices.enabled = false
```

```
quarkus.keycloak.devservices.enabled = false
```

quarkus.[dev_service].devservices.image-name

quarkus.datasource.devservices.image-name = registry.example.com/my-user/mysql:8

Quarkus test resources instead of Dev Services

```
@QuarkusTest
@QuarkusTestResource(CustomBrokerResource.class)
public class MyTest {

    @Test
    public someTest() {
        // Perform the test that uses your external service
    }
}
```

Quarkus test resources instead of Dev Services

```
public class CustomBrokerResource (1)
    implements QuarkusTestResourceLifecycleManager {

    DockerImageName IMAGE = DockerImageName.parse(
        "quay.io/myorg/custom-broker:v1.0"); (2)

    GenericContainer<?> SERVICE = new GenericContainer<>( IMAGE )
       .withExposedPorts(9092)
       .withEnv("API_URL", "localhost:9092"); (3)

    @Override
    public Map<String, String> start() { (4)
        SERVICE.start(); (5)

        return Map.of("broker.internal.port", SERVICE.getPort()); (6)
    }

    @Override
    public void stop() { (7)
        SERVICE.stop(); (8)
    }
}
```

Quarkus test resources instead of Dev Services

```
@QuarkusTestResource( 1
    CustomBrokerResource.class 2
)
@Retention(RetentionPolicy.RUNTIME) 3
@Target(ElementType.TYPE) 4
public @interface WithCustomBroker { 5
    String myPort() default ""; 6 7
}
```

Quarkus test resources instead of Dev Services

```
public class CustomBrokerResource implements
QuarkusTestResourceConfigurableLifecycleManager<WithCustomBroker> { (1)(2)

    DockerImageName IMAGE = DockerImageName.parse(
        "quay.io/myorg/custom-broker:v1.0");
    GenericContainer<?> SERVICE = new GenericContainer<>( IMAGE );
    private String port;

    @Override
    public void init( WithCustomBroker params ) { (3)
        port = params.myPort(); (4)
    }

    @Override
    public Map<String, String> start() {
        SERVICE
            .withExposedPorts( port )
            .start();
         return Map.of("some.custom.broker.port", port );
     }

     @Override
     public void stop() {
         SERVICE.stop();
     }
}
```

```
@QuarkusTest
@WithMyCustomBroker( myPort = "9090" )
public class MyTest {

    @Test
    public someTest() {
        // Perform the test that uses your external service
    }
}
```

Recursos

https://www.testcontainers.org/

https://www.testcontainers.org/supported_docker_environment/

https://quarkus.io/version/2.13/guides/dev-services