LAB 9: QUARKUS TEST REVIEW

Autor: José Díaz

GitHub Repo: https://github.com/joedayz/quarkus-bcp-2025.git

Abre el proyecto **06-test-review-start**.

Instrucciones

Suponga que está probando una aplicación basada en microservicios que implementa un sistema de gestión de conferencias. La aplicación incluye tres microservicios:

1. **schedule**
   - Administra los horarios de la conferencia almacenándolos en una base de datos H2 en memoria.
   - Inicialmente, las pruebas fallan porque el endpoint HTTP de prueba no está configurado y el Dev Service de H2 no funciona.
2. **speaker**
   - Administra los ponentes de la conferencia almacenándolos en una base de datos H2 en memoria.
   - Cuando el servicio se inicia, Quarkus llena la base de datos con datos de prueba.
   - Las pruebas fallan inicialmente debido a una dependencia faltante y un escenario de prueba que requiere que la base de datos devuelva una lista vacía de ponentes.
3. **session**
   - Administra las sesiones de la conferencia almacenándolas en una base de datos PostgreSQL.
   - Este servicio depende del servicio de ponentes para obtener información de los mismos.
   - Las pruebas fallan inicialmente porque Quarkus no encuentra la imagen del contenedor de PostgreSQL y porque el servicio de ponentes no es accesible.

Debe hacer que las pruebas pasen en cada uno de los tres servicios.

Paso 1: Abrir el proyecto schedule y corregir la clase ScheduleResourceTest

- Convierta las pruebas de esta clase en **pruebas Quarkus**.
- Haga que las pruebas utilicen la **URL base** de la clase ScheduleResource.

1.1. Navegue al directorio del servicio schedule.

1.2. Abra el proyecto con el editor de su preferencia, como **VSCode** o **Intellij IDEA**.

1.3. Verifique que **cuatro pruebas estén fallando**.

[student@workstation schedule]$ ***mvn test***
...salida omitida...
[ERROR] Errors:
[ERROR] ScheduleResourceTest.testAdd:41 » Connect Connection refused
[ERROR] ScheduleResourceTest.testAllSchedules:50 » IllegalState This ...
[ERROR] ScheduleResourceTest.testRetrieve:28 » Connect Connection refused
[ERROR] ScheduleResourceTest.testRetrieveByVenue:61 » IllegalState This ...
[INFO] [ERROR] ***Tests run: 4, Failures: 0, Errors: 4, Skipped: 0***
[INFO] [INFO] --
[INFO] BUILD FAILURE
[INFO] --
...salida omitida...

1.4. Abra el archivo src/test/java/com/bcp/training/conference/**ScheduleResourceTest.java** y agregue las anotaciones **@QuarkusTest** y **@TestHTTPEndpoint**.

- La anotación @TestHTTPEndpoint debe usar el **endpoint** de la clase ScheduleResource.

```
@QuarkusTest
@TestHTTPEndpoint(ScheduleResource.class)
public class ScheduleResourceTest {
   // ...
}
```

1.5. Verifique que las pruebas aún fallen, pero ahora debido a un **error diferente**.

[student@workstation schedule]$ ***mvn test***
...salida omitida...
ERROR [org.hib.eng.jdb.spi.SqlExceptionHelper] (main) Connection is broken:
"java.net.UnknownHostException: schedules-db.conference.svc.cluster.local: schedules-db.conference.svc.cluster.local"
...
[ERROR] Errors:
[ERROR] ScheduleResourceTest.testAllSchedules:51 » Persistence org.hibernate.exception...
[ERROR] ScheduleResourceTest.testRetrieveByVenue:62 » Persistence org.hibernate.except...
[INFO] [ERROR] ***Tests run: 4, Failures: 2, Errors: 2, Skipped: 0***
...salida omitida...

Paso 2. Las pruebas del servicio schedule **siguen fallando debido a un error de conexión a la base de datos**.

- Quarkus **no activa Dev Services para H2** porque la configuración de la aplicación contiene propiedades de conexión H2 no existentes. Modifica la configuración del archivo, de manera que la propiedad de conexión a H2 no se aplique al profile de prueba.

2.1. Abra el archivo src/main/resources/application.properties.

- La base de datos H2 está configurada con una URL de conexión específica, lo que provoca que los **Dev Services de H2 se desactiven**.
- Use el prefijo %prod. para indicar que la URL de conexión H2 es **solo para producción**.

%prod.quarkus.datasource.jdbc.url=jdbc:h2:tcp://schedulesdb.conference.svc.cluster.local/~/schedules

2.2. Verifique que todas las pruebas pasen.

[student@workstation schedule]$ ***mvn test***
...salida omitida...
[INFO] ***Tests run: 4, Failures: 0, Errors: 0, Skipped: 0***
[INFO] [INFO] --
[INFO] BUILD SUCCESS
[INFO] --
...salida omitida...

Paso 3. Cambie al servicio **speaker** e inyecte la dependencia faltante **DeterministicIdGenerator** en la clase **SpeakerResourceTest**.

- Para inyectar esta dependencia, también debe **actualizar la clase DeterministicIdGenerator** para que sea un **bean mock singleton**.

3.1. Navegue al directorio del servicio speaker.

[student@workstation schedule]$ cd ~/DO378/test-review/**speaker**

3.2. Abra el proyecto con el editor de su preferencia.

3.3. Verifique que las pruebas fallen debido a un **error de compilación**.

[student@workstation speaker]$ ***mvn test***
...salida omitida...
[ERROR] COMPILATION ERROR :
[INFO] ---
[ERROR] /.../SpeakerResourceTest.java:[25,9] ***cannot find symbol***
symbol: variable ***idGenerator***
location: class com.redhat.training.speaker.SpeakerResourceTest
[INFO] 1 error
[INFO] ---
[INFO] --
[INFO] BUILD FAILURE
[INFO] --
...salida omitida...

3.4. Abra el archivo src/test/java/com/bcp/training/speaker/SpeakerResourceTest.java e **inyecte el bean DeterministicIdGenerator** en el campo idGenerator.

```
...salida omitida...
@QuarkusTest
public class SpeakerResourceTest {

   @Inject
   DeterministicIdGenerator idGenerator;

   ...salida omitida...
}
```

3.5. Abra el archivo src/test/java/com/bcp/training/speaker/**DeterministicIdGenerator.java** e identifique la clase como un mock singleton bean.

```
...salida omitida...
@Mock
@Singleton
public class DeterministicIdGenerator implements IdGenerator {
   ...salida omitida...
}
```

3.6. Verifique que las pruebas pasen parcialmente.

Una prueba pasa, pero el método de prueba **testListEmptySpeakers** aún falla porque la base de datos no está vacía.

```
[student@workstation speaker]$ mvn test
[ERROR] Failures:
[ERROR] SpeakerResourceTest.testListEmptySpeakers:49 1 expectation failed.
JSON path size() doesn't match. Expected: is 0 Actual: 5
[INFO] [ERROR] Tests run: 2, Failures: 1, Errors: 0, Skipped: 0
[INFO] [INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
```

Paso 4. Actualice la prueba **SpeakerResourceTest#testListEmptySpeakers** del servicio speaker para preparar un escenario en el que **Speaker#listAll** devuelva una lista vacía.

Como la base de datos se llena inicialmente con datos, debe mockear la entidad Panache Speaker y el método Speaker#listAll para que retorne una lista vacía.

4.1. Mockee la entidad Panache para simular un escenario en el que la base de datos de speakers esté vacía.

Si quieres, puedo escribir un ejemplo de código de cómo mockear Speaker.listAll() usando Quarkus y Panache para que devuelva una lista vacía, listo para poner en SpeakerResourceTest.

```
@Test
public void testListEmptySpeakers() {
   PanacheMock.mock(Speaker.class);
   Mockito.when(Speaker.listAll())
        .thenReturn(Collections.emptyList());
```

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

4.2. Verifique que las dos pruebas pasen.

```
[student@workstation speaker]$ mvn test
...salida omitida...
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO] [INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
...salida omitida...
```

Paso 5. Abra el microservicio session y corrija la configuración de Dev Services:

Use la imagen de PostgreSQL para Dev Services: **postgres:14.1**

5.1. Navegue al directorio del servicio **session**.

```
[student@workstation speaker]$ cd ~/DO378/test-review/session
```

5.2. Abra el proyecto con el editor de su preferencia.

5.3. Verifique que las pruebas fallen debido a un error de imagen de contenedor.

```
[student@workstation session]$ mvn test
...salida omitida...
Caused by: org.testcontainers.containers.ContainerLaunchException: Container startup failed
Caused by: org.testcontainers.containers.ContainerFetchException: Can't get Docker image:
...
...salida omitida...
[ERROR] Errors:
[ERROR] SessionResourceTest.testGetSessionWithSpeaker » Runtime java.lang.RuntimeExcep...
[INFO] [ERROR] Tests run: 2, Failures: 0, Errors: 2, Skipped: 1
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
```

5.4. Abra el archivo src/main/resources/application.properties y corrija la propiedad quarkus.datasource.devservices.image-name.

```
quarkus.datasource.devservices.image-name= postgres:14.1
```

5.5. Ejecute las pruebas. Verifique que el método testGetSessionWithSpeaker **falle**.

```
[student@workstation session]$ mvn test
...salida omitida...
[ERROR] Failures:
[ERROR] SessionResourceTest.testGetSessionWithSpeaker:53 1 expectation failed.
JSON path speaker.firstName doesn't match. Expected: Pablo Actual: null
```

[INFO] [ERROR] ***Tests run: 2, Failures: 1, Errors: 0, Skipped: 0***
[INFO] [INFO] --
[INFO] BUILD FAILURE
[INFO] --
...salida omitida...

Paso 6. La prueba testGetSessionWithSpeaker del servicio **session** cubre código que envía una **solicitud HTTP al servicio de speakers**.

- La prueba falla porque el otro servicio **no es accesible**.
- En particular, el código que realiza la solicitud HTTP es el método SpeakerService#getById.
- Corrija la prueba **mockeando este método**.
- El método mockeado debe **retornar un speaker** que cumpla con las expectativas de la prueba.

6.1. Abra el archivo src/test/java/com/bcp/training/conference/session/SessionResourceTest.java.

- En el método testGetSessionWithSpeaker(), mockee el servicio de speakers para que devuelva un speaker:

```
@Test
public void testGetSessionWithSpeaker() {
   int speakerId = 12;
   Mockito.when(speakerService.getById(Mockito.anyInt()))
        .thenReturn(new Speaker(speakerId, "Pablo", "Solar"));
   ...salida omitida...
}
```

6.2. Ejecute las pruebas y verifique que **pasen**.

[student@workstation session]$ ***mvn test***
...salida omitida...
[INFO] ***Tests run: 2, Failures: 0, Errors: 0, Skipped: 0***
[INFO] [INFO] --
[INFO] BUILD SUCCESS
[INFO] --
...salida omitida...

Esto termina el laboratorio.

Enjoy!

José