Ejercicios de DbUnit

Índice

1. Generar el XML de DbUnit (0.5 puntos)

Vamos añadir tests de DbUnit al proyecto de la filmoteca para probar los métodos del `JDBCPeliculaDAO`. Copiamos los .jar proporcionados en las plantillas de la sesión a la carpeta `lib` del proyecto y los importamos en el proyecto. Para comprobar que funcionan añadimos al proyecto el archivo `TestDataExtractor.java` y lo ejecutamos. Debe generar un archivo XML como el siguiente:

```
<?xml version='1.0' encoding='UTF-8'?>
<dataset>
  <peliculas id="1" titulo="El Resplandor"/>
  <peliculas id="3" titulo="Casablanca"/>
  <peliculas id="5" titulo="Borrame facil"/>
  <directores id="1" nombre="Michael Curtiz"/>
  <directores id="2" nombre="Stanley Kubrick"/>
  <peliculas_directores pelicula_id="1" director_id="1"/>
  <peliculas_directores pelicula_id="3" director_id="2"/>
</dataset>
```

Podemos guardar este archivo como `resources/db-init.xml` para utilizarlo después como punto de partida de los tests de DBUnit.

2. Clase para los test de DbUnit (1 punto)

En la carpeta de fuentes `test`, en el paquete `es.ua.jtech.lja.filmoteca.dao` creamos una nueva clase llamada `TestJDBCPeliculaDAO`. En sus métodos utilizaremos el `JDBCPeliculaDAO` y un objeto `IDatabaseTester` así que los declaramos como campos privados de la clase.

Inicializaremos dichos campos en el método `@Before public void setUp()`. En el mismo método inicializaremos el dataset de la base de datos para empezar siempre los tests desde el mismo estado:

```
FlatXmlDataSetBuilder builder = new FlatXmlDataSetBuilder();
IDataSet dataSet =
builder.build(this.getClass().getResourceAsStream("/db-init.xml"));
databaseTester.setDataSet(dataSet);
```

La última operación del método `setUp()` será la llamada a `.onSetup()` del objeto de tipo `IDatabaseTester`.

A su vez, el método `@After public void tearDown()` llamará a `.onTearDown()` del objeto de tipo `IDatabaseTester`.

3. Test de borrado y añadido (0.5 puntos)

Los test deben llevar la anotacion `@Test`. Crear el primero de ellos llamado

`test1DelAdd()`. Este test deberá utilizar el DAO obtener todas las películas en una `List`, después recorrerlas en ese mismo orden eliminándolas de la base de datos y finalmente insertarlas una a una también en el mismo orden.

Conectar a la base de datos con el `IDatabaseConnection` para crear a partir de la conexión el `QueryDataSet` con las tablas películas, directores y películas_directores. Una vez creado el set de datos, escribirlo en un nuevo fichero `db-output-data.xml`. Comparar utilizando `Assert.assertEquals( )` el XML generado con el del archivo `db-expected1.xml` que deberá estar en la carpeta de recursos y contener exactamente lo mismo que el `db-init.xml`.

4. Test que espera una excepción (0.5 puntos)

Crear el segundo test llamado `test2AddExisting()`. Este test deberá utilizar el DAO para añadir una película existente. Se puede proceder obteniendo la lista de películas del DAO e intentando insertar una de ellas. Según la especificación, deberá saltar una `DAOException`. Utilizar la notación

```
@Test(expected=DAOException.class)
```

para que el test falle si no se produce exactamente esa excepción.

5. Test de borrado (0.5 puntos)

Crear el tercer test llamado `test3Del()`. Este test deberá utilizar el DAO para eliminar la película con identificador `id==4`. El XML esperado se llamará `db-expected3.xml` y deberá contener todas las películas iniciales menos la 4.

Al ejecutar los tests como JUnit deberán dar luz verde.