Ejercicio (30 min.)

# EXTERNAL DATA SOURCE y NUKE AND PAVE

1. Del ejercicio anterior “EmployeeJDBCTest\_SelfContained\_Transaction”, elegir una de las pruebas implementadas para el método “find” y copiarla completamente a este ejercicio. Luego de copiar la prueba, eliminar las líneas que contienen el método LoadData.  
     
   Con esto ya tenemos el Arrange-Act-Assert de nuestra prueba pero nos falta un mecanismo para inicializar y restablecer datos en la BD.
2. INICIALIZAR LA BD (EXTERNAL DATA SOURCE)

Para inicializar la BD vamos a utilizar DBUnit, esta herramienta nos permite insertar datos que se encuentran en un archivo externo XML. Para esto realizar los siguientes pasos:

1. INSERTAR LOS DATOS DEL ARCHIVO EN LA BD

Como los datos van a ser inicializados cada vez que se ejecute una prueba, vamos a colocar la inicialización dentro de un método “SetUp”, creamos este método encima de nuestro método de prueba.

@Before

**public** **void** loadData() **throws** Exception{

}

Utilizando BDUnit cargarmos los datos en la BD. Par esto colocar el siguiente fragmento de código dentro del método “loadData()”.

InputStream file = **this**.getClass().getResourceAsStream("employeeDataSet.xml");

IDataSet dataSet = **new** FlatXmlDataSetBuilder().build(file);

IDatabaseConnection connection = **new** DatabaseConnection(dataSource.getConnection());  
DatabaseOperation.*INSERT*.execute(connection, dataSet);

PREGUNTA: ¿Se comprende qué es lo que hace cada una de estas líneas de código?

Existen 3 maneras básicas de insertar datos:

* DatabaseOperation.INSERT.execute(connection, dataSet);
* DatabaseOperation.CLEAN\_INSERT.execute(connection, dataSet);
* new InsertIdentityOperation(DatabaseOperation.INSERT).execute(connection,dataSet);

PREGUNTAS:

* CLEAN\_INSERT elimina todos los datos de la tabla antes de insertar, ¿Por qué que en esta oportunidad solo utilizamos INSERT? ¿En qué circunstancias utilizaríamos CLEAN\_INSERT?
* En nuestra inicialización utilizamos InsertIdentityOperation(DatabaseOperation.INSERT), ¿Cuál es crees que sea la diferencia entre InsertIdentityOperation(DatabaseOperation.INSERT) y DatabaseOperation.INSERT? (La respuesta está relacionada con la columna ID de la tabla de BD).

1. CREAR EL ARCHIVO XML CON LOS DATOS

Abrir el archivo src/test/resources/testautomation/databasetesting/jdbc/employeeDataSet.xml, examinar su contenido y estructura.

<dataset>

<employees lastName=*"Pacheco"* hireDate=*"2012-12-28"*/>

<employees lastName=*"Quispe"* hireDate=*"2012-12-29"*/>

<employees lastName=*"Tovar"* hireDate=*"2012-12-30"*/>

</dataset>

¿Se percataron que tiene los mismos nombres que la tabla y columnas de la BD?

1. REESTABLECER LA BD

Como los datos van a ser limpiados cada vez que se finalice una prueba, vamos a colocar la eliminación dentro de un método “TearDown”, creamos este método encima de nuestro método de prueba.

@After

**public** **void** cleanData() **throws** Exception{

}

Utilizando BDUnit cargar los datos en la BD. Par esto colocar el siguiente fragmento de código dentro del método “cleanData()”.

InputStream file = **this**.getClass().getResourceAsStream("employeeDataSet.xml");

IDataSet dataSet = **new** FlatXmlDataSetBuilder().build(file);

IDatabaseConnection connection = **new** DatabaseConnection(dataSource.getConnection());

DatabaseOperation.*DELETE\_ALL*.execute(connection, dataSet);

PREGUNTA: ¿Se comprende qué es lo que hace cada una de estas líneas de código?

Existen 2 maneras básicas de eliminar datos:

* DatabaseOperation.DELETE\_ALL.execute(connection, dataSet);
* DatabaseOperation.DELETE.execute(connection, dataSet);

PREGUNTA:

* DELETE\_ALL elimina todos los datos de las tablas que se encuentra definidos en el “dataset”, ¿Cuál crees que sea la diferencia entre DELETE\_ALL y DELETE?

1. Revisar las dependencias y el bean transactionmanager.
2. Implementar la segunda prueba para el método “find”.

La idea es agregar más registros a la fuente de datos de tal manera que podamos compartirla para toda la prueba.

1. Implementa la prueba para el “créate”.
2. Implementa la prueba para el “delete”.

Cambiar la operación por:   
new InsertIdentityOperation(DatabaseOperation.INSERT).execute(connection,dataSet);

<dataset>

<employees id=*"1"* lastName=*"Pacheco"* hireDate=*"2012-12-28"*/>

<employees id=*"2"* lastName=*"Quispe"* hireDate=*"2012-12-29"*/>

<employees id=*"3"* lastName=*"Tovar"* hireDate=*"2012-12-30"*/>

</dataset>

* Problemas de los diferentes tipos de ids: Identity, Sequence, etc
* No siempre es necesario insertar el Id, depende de la prueba.

NOTA: Es preferible utilizar SelContained, utilizar DBUnit cuando tengamos que cargar una cantidad importante de datos ya que es difícil mantener los archivos. Otro ejemplo de su uso es pruebas de sistema.