

¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
 Ese apoyo que siempre quiso tener...

1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,
 HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
 Gestor de contenidos (Alfresco)
 Aplicaciones híbridas

Tareas programadas (Quartz)
 Gestor documental (Alfresco)
 Inversión de control (Spring)

Control de autenticación y
 acceso (Spring Security)
 UDDI
 Web Services
 Rest Services
 Social SSO
 SSO (Cas)

JPA-Hibernate, MyBatis
 Motor de búsqueda empresarial (Solr)
 ETL (Talend)

Dirección de Proyectos Informáticos.
 Metodologías ágiles
 Patrones de diseño
 TDD

BPM (jBPM o Bonita)
 Generación de informes (JasperReport)
 ESB (Open ESB)



Inicio Quienes somos Tutoriales Formación Comparador de salarios Comic Charlas Más

Estas en: Inicio Tutoriales DBUnit-Exportar e Importar BBDD

Últimas Noticias

- » Lanzamiento del nuevo Web de Autentia
- » Historia de la Informática. Capítulo 72. 1995 (1ª Parte)
- » Historia de la Informática. Capítulo 71. 1994
- » Historia de la Informática. Capítulo 70. 1993
- » Si se pregunta ¿Qué ofrece este Web?
- » Autentia en la Sun Open Communities Forum
- » Autentia cumple 6 años
- » Comentario del libro: El economista naturalista de Robert Frank

+Noticias Destacadas

- » Lanzamiento del nuevo Web de Autentia
- » Contratos ágiles: Vendiendo Scrum a tus clientes.
- » Quinta charla Autentia + Proyectalis + Agile Spain: Contratos ágiles: Vendiendo Scrum a tus clientes
- » Lo mejor de esta semana: Curso de Scrum con Ángel Medinilla

+Comentarios Cómic

+Enlaces

Catálogo de servicios Autentia (PDF 6,2MB)



En formato comic...

Tutorial desarrollado por



Saúl García Díaz

Consultor tecnológico de desarrollo de proyectos de informática.

Puedes encontrarme en Autentia

Somos expertos en Java/JEE

Catálogo de servicios de Autentia

Descargar (6,2 MB)

Descargar en versión comic (17 MB)

AdictosALTrabajo.com es el Web de difusión de conocimiento de Autentia.



Catálogo de cursos

Descargar este documento en formato PDF: [DBUnitEI.pdf](#)

Fecha de creación del tutorial: 2009-07-06

DBUnit - Exportar e Importar BBDD

0. Índice de contenidos.

- 1. Entorno.
- 2. Introducción
- 3. DBUnit-Definición
- 4. DBUnit-Componentes
- 5. Clase DBUnitUtils
- 6. Exportando BBDD
- 7. Importando BBDD
- 8. Borrando BBDD
- 9. Test JUnit
- 10. Conclusiones.

1. Entorno

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Dell Latitude E5500(Core Duo T9550 2.66GHz, 4GB RAM, 340 GB HD)
- Sistema operativo: Windows XP.
- Eclipse ganymede
- Maven 2.1.0
- JUnit 4.4
- Spring 2.5.6
- PostgreSQL 8.2
- Hsqldb 1.8.0.7

2. Introducción

Uno de los puntos importantes dentro de todo desarrollo debería de ser los test unitarios. Con ellos podemos garantizar que nuestro código hace realmente lo que esperamos que haga y, aunque parezca una obviedad, no siempre es así. Para probar nuestras aplicaciones Java disponemos de un excelente framework como JUnit, que permite evaluar si cada uno de los métodos de nuestras clases se comporta como se espera, es decir, en función de los valores de entrada se evalúa un retorno esperado. Pero, ¿qué ocurre cuando nuestros test necesitan interactuar con una base de datos?, aquí es donde entra en juego DBUnit proporcionándonos las herramientas necesarias para exportar e importar nuestra base de datos de una manera sencilla.

3. DBUnit-Definición

DBUnit es una extensión de JUnit para realizar test unitarios que permiten interactuar con un conjunto de datos. También permite dejar la base de datos en un estado conocido antes y después de cada test, esto con el fin de prevenir que datos corruptos productos de test fallidos queden en la base de datos ocasionando problemas a los test siguientes.

4. DBUnit-Componentes

Los componentes principales son:

- IDatabaseConnection. Interfaz que representa una conexión DBUnit a la base de datos.
- IDataSet. Interfaz que representa una colección de tablas (Manipula tablas y datos).
- DatabaseOperation. Clase abstracta que representa la operación que se va a realizar sobre la base de datos antes o después de un test.



Web

www.adictosaltrabajo.com

Buscar

Últimos tutoriales

2009-08-20
[Selenium IDE-Incorporando while en los test](#)

2009-08-14
[Blender y JMonkeyEngine. Exportación de archivos Blender y uso de los mismos en JMonkeyEngine](#)

2009-08-14
[5º tutorial TNT Concept Versión 0.16.1 Gestión de informes, vacaciones y utilidades](#)

2009-08-14
[Joomla 1.5. Instalación y configuración](#)

2009-08-13
[Introducción a los diagramas EPC \(Event-Driven Process Chain\)](#)

2009-08-10
[Blender. Animaciones avanzadas y renderización](#)

2009-08-10
[Gestión de Calidad, tablón y seguimiento en TNT Concept Versión 0.16.1](#)

2009-08-10
[Cómo hacer una página web](#)

2009-08-06
[Tips And Tricks JUnit Spring](#)

2009-08-03
[Instalación de VirtualBox PUEL](#)

2009-08-03
[Gestión de contactos y pedidos en TNT Concept versión 0.16.1](#)

De estos componentes cabe destacar:

IDataset:

- FlatXmlDataSet. Dataset leído y escrito como un documento XML planos.
- XmlDataSet. Dataset leído y escrito como un documento XML (DTD).
- DatabaseDataSet. Adaptador que provee acceso a la BD mediante una instancia de un dataset.
- QueryDataSet. Mantiene la colección de tablas resultado de una consulta a la BD en un dataset.
- DefaultDataSet. Usado para crear datasets de manera programada.
- XlsDataSet. Dataset leído y escrito.

DatabaseOperation:

- DatabaseOperation.UPDATE. Actualiza la BD con la información del dataset.
- DatabaseOperation.INSERT. Inserta en la BD el contenido del dataset.
- DatabaseOperation.DELETE. Elimina de la BD solamente los datos especificados en el dataset.
- DatabaseOperation.DELETE_ALL. Elimina todas las filas de las tablas especificadas en el dataset.
- DatabaseOperation.TRUNCATE. Hace un truncate de las tablas especificadas en el dataset.
- DatabaseOperation.REFRESH. Esta operación refresca el contenido del dataset en la BD, es decir, si es el caso actualiza o inserta nuevos registros.
- DatabaseOperation.CLEAN_INSERT. DELETE_ALL + INSERT.
- DatabaseOperation.NONE. No hace nada.

5. Clase DBUnitUtils

Esta es la clase de la que nos serviremos para exportar e importar nuestros juegos de datos con el fin de complementar los test unitarios. A lo largo del tutorial explicaremos cada una de las partes de esta clase con el fin de aprender el manejo básico de DBUnit.

```

view plain copy to clipboard print ?
01. package com.autentia.dbunit.test;
02.
03. import java.io.FileInputStream;
04. import java.io.FileOutputStream;
05. import java.sql.Connection;
06. import java.sql.Driver;
07. import java.sql.DriverManager;
08. import java.sql.SQLException;
09. import org.dbunit.database.DatabaseConnection;
10. import org.dbunit.database.DatabaseSequenceFilter;
11. import org.dbunit.database.IDatabaseConnection;
12. import org.dbunit.database.QueryDataSet;
13. import org.dbunit.dataset.FilteredDataSet;
14. import org.dbunit.dataset.IDataset;
15. import org.dbunit.dataset.xml.FlatXmlDataSet;
16. import org.dbunit.dataset.xml.FlatXmlWriter;
17. import org.dbunit.operation.DatabaseOperation;
18.
19. public class DBUnitUtils
20. {
21.
22.
23.     public static void generateXML(String driverName,
24.                                   String urlDB,
25.                                   String userDB,
26.                                   String passwordDB,
27.                                   String schemaDB,
28.                                   String nameXML) throws SQLException {
29.
30.         Connection conn=null;
31.
32.         try {
33.             // Connect to the database
34.             DriverManager.registerDriver((Driver)Class.forName(driverName).newInstance());
35.             conn = DriverManager.getConnection(urlDB, userDB, passwordDB);
36.             IDatabaseConnection connection = new DatabaseConnection(conn, schemaDB);
37.
38.
39.             DatabaseSequenceFilter filter = new DatabaseSequenceFilter(connection);
40.             IDataset datasetAll = new FilteredDataSet(filter, connection.createDataSet());
41.             QueryDataSet partialDataSet = new QueryDataSet(connection);
42.
43.             String[] listTableNames = filter.getTableNames(datasetAll);
44.             for (int i = 0; i < listTableNames.length; i++) {
45.                 final String tableName = listTableNames[i];
46.                 // Specify the SQL to run to retrieve the data
47.                 partialDataSet.addTable(tableName);
48.
49.             }
50.
51.             // Specify the location of the flat file(XML)
52.             FlatXmlWriter datasetWriter = new FlatXmlWriter(new FileOutputStream
("C:\\\\" + nameXML + ".xml"));
53.
54.             // Export the data
55.             datasetWriter.write(partialDataSet);
56.
57.             } catch (Exception exc) {
58.                 exc.printStackTrace();
59.
60.             } finally{
61.                 conn.close();
62.             }
63.         }
64.
65.     public static void generatePartialXML(String driverName,
66.                                           String urlDB,
67.                                           String userDB,
68.                                           String passwordDB,
69.                                           String schemaDB,
70.                                           String nameXML) throws SQLException {
71.
72.         Connection conn=null;
73.         try {

```

2009-08-03
Comentando el libro: La estrategia del océano azul

2009-07-30
Funciones esenciales para crear un juego.

2009-07-30
2º tutorial TNT Concept versión 1.16.1

2009-07-29
Hibernate Search, Bridges, Analizadores y más

2009-07-24
Migración de EJB3 a JPA y Spring.

2009-07-20
Directorio de ejemplos de JMonkey Engine

2009-07-19
JSR-179 Location API para J2ME: Posicionamiento geográfico en nuestras aplicaciones.

2009-07-16
Gestión de Usuarios en TNT Concept versión 0.16.1

2009-07-16
Continuación del Tutorial: JMonkeyEngine, Creación de nuestro primer juego.

2009-07-16
Como implementar el Scene Monitor para analizar las escenas en JMonkeyEngine

2009-02-26
Transformaciones de escena en JMonkeyEngine

2009-07-15
Detalles del juego de la moto en jMonkeyEngine.

2009-07-14
JMonkeyEngine, Creación de nuestro primer juego.

2009-07-13
Ajax tests con Selenium: prototype.js e ICEfaces.

2009-07-08
AOP con AspectJ y Maven

2009-07-07
Instalación y configuración de Eclipse Galileo

2009-07-07
Iniciarse en el manejo de JME, Creación de un Cloth.

2009-07-06
Primeros pasos con Blender: Pintando nuestra mascota en 3D

2009-07-06
DBUnit-Exportar e Importar BBDD

2009-07-05
JMeter, Pruebas de stress

```

74.         // Connect to the database
75.         DriverManager.registerDriver((Driver)Class.forName(driverName).newInstance());
76.         conn = DriverManager.getConnection(urlDB, userDB, passwordDB);
77.         IDatabaseConnection connection = new DatabaseConnection(conn, schemaBD);
78.
79.         QueryDataSet partialDataSet = new QueryDataSet(connection);
80.         // Specify the SQL to run to retrieve the data
81.         partialDataSet.addTable("web_direccion");
82.         partialDataSet.addTable("web_usuario");
83.
84.         // Specify the location of the flat file(XML)
85.         FlatXmlWriter datasetWriter = new FlatXmlWriter(new FileOutputStream
("C:\\\" + nameXML + ".xml"));
86.
87.         // Export the data
88.         datasetWriter.write(partialDataSet);
89.
90.     } catch (Exception exc) {
91.         exc.printStackTrace();
92.     } finally{
93.         conn.close();
94.     }
95. }
96.
97. public static void createData(String driverName,
98.                               String urlDB,
99.                               String userDB,
100.                              String passwordDB,
101.                              String nameXML) throws SQLException {
102.
103.     Connection conn = null;
104.     try {
105.         // Connect to the database
106.         DriverManager.registerDriver((Driver)Class.forName(driverName).newInstance());
107.         conn = DriverManager.getConnection(urlDB, userDB, passwordDB);
108.         IDatabaseConnection connection = new DatabaseConnection(conn);
109.
110.         DatabaseOperation.INSERT.execute(connection, new FlatXmlDataSet(new FileInputStream
("C:\\\" + nameXML + ".xml")));
111.
112.     } catch (Exception exc) {
113.         exc.printStackTrace();
114.     } finally{
115.         conn.close();
116.     }
117. }
118.
119. public static void deleteData(String driverName,
120.                               String urlDB,
121.                               String userDB,
122.                               String passwordDB,
123.                               String nameXML) throws SQLException {
124.
125.     Connection conn = null;
126.     try {
127.         // Connect to the database
128.         DriverManager.registerDriver((Driver)Class.forName(driverName).newInstance());
129.         conn = DriverManager.getConnection(urlDB, userDB, passwordDB);
130.         IDatabaseConnection connection = new DatabaseConnection(conn);
131.         DatabaseOperation.DELETE.execute(connection, new FlatXmlDataSet(new FileInputStream
("C:\\\" + nameXML + ".xml")));
132.
133.     } catch (Exception exc) {
134.         exc.printStackTrace();
135.     } finally{
136.         conn.close();
137.     }
138. }
139. }

```

6. Exportando BBDD

En esta sección veremos como con DBUnit es posible generar un XML a partir de una base de datos existente de una manera sencilla; pero antes hemos de tener en cuenta algunas consideraciones:

- Usar una instancia de base de datos por desarrollador.
- Usar datasets pequeños (múltiples).
 - Según la documentación oficial a partir de la versión 2.0 es posible usar datasets de gran tamaño pero en realidad el uso de grandes datasets provoca problemas de memoria.
 - Este punto es importante, pero la decisión dependerá en gran medida del volumen de datos que estemos manejando, es decir, si el volumen de datos es muy elevado es muy aconsejable trabajar con dataset parciales; si no es así, podríamos trabajar con dataset completos sin problemas.
- Realizar la configuración de la carga de datos una sola vez por clase de test o test suite.

6.1 Exportación parcial

La implementación de este tipo de solución dependerá de la necesidad de cada momento. Por ejemplo:

- Necesitamos los datos de un par de tablas. Nuestro método quedaría algo como:

```

view plain copy to clipboard print ?
01. ....
02. ....
03.
04. public static void generatePartialXML(String driverName,
05.                                     String urlDB,
06.                                     String userDB,
07.                                     String passwordDB,
08.                                     String schemaBD,
09.                                     String nameXML) throws SQLException {
10.
11.     Connection conn=null;
12.
13.     try {
14.         // Connect to the database
15.         DriverManager.registerDriver((Driver)Class.forName(driverName).newInstance());

```

sobre aplicaciones web:
Grabando y reproduciendo navegaciones

2009-07-02
Axis2: Invocación de Servicios Web usando distintos MEP

2009-07-02
Instalación OpenOffice

2009-07-02
Juegos 3D en Java: Blender y JMonkeyEngine

2009-06-20
StAX (Xml Pull Parser): Streaming API para XML

2009-06-15
Configuración de la desconexión de usuarios con ICEFaces

2009-06-10
LWUIT: Una librería gráfica tipo AWT o Swing para J2ME

2009-06-10
Mapas mentales con XMind

2009-02-26
Redimensionar Imágenes en Windows Vista

2009-06-08
UploadFile con Icefaces + Hibernate + Anotaciones

2009-06-05
Habilitar exportación en Liferay

2009-06-01
Registrar Liferay en Eclipse

2009-05-29
Liferay Social Office

2009-05-28
Broadcast con Ustream

2009-05-25
Tabla datos accesible con ordenación y paginación

2009-05-21
Primeros pasos con Audacity: Un editor de sonido libre y multiplataforma.

2009-05-11
Introducción a TortoiseSVN

2009-05-07
Hacer 'scp' de varios ficheros sin solicitud de clave

2009-05-02
Plugin Hibernate3 para Maven

2009-04-26
AgileDraw: una técnica rápida de modelado

```

15.         conn = DriverManager.getConnection(urlDB, userDB, passwordDB);
16.         IDatabaseConnection connection = new DatabaseConnection(conn, schemaBD);
17.
18.         QueryDataSet partialDataSet = new QueryDataSet(connection);
19.         // Specify the SQL to run to retrieve the data
20.         partialDataSet.addTable("web_direccion");
21.         partialDataSet.addTable("web_usuario");
22.
23.         // Specify the location of the flat file(XML)
24.         FlatXmlWriter datasetWriter = new FlatXmlWriter(new FileOutputStream
("C:\\\" + nameXML + ".xml"));
25.
26.         // Export the data
27.         datasetWriter.write(partialDataSet);
28.
29.     } catch (Exception exc) {
30.         exc.printStackTrace();
31.     } finally{
32.         conn.close();
33.     }
34. }
35.
36. ....
37. ....

```

Este es el caso más sencillo que nos podemos encontrar, ya que nuestro test necesita un conjunto de datos obtenidos de un número reducido de tablas y, además, estas tablas no tienen relación entre sí. Sin embargo, debemos prestar especial atención a las relaciones entre las tablas que queremos exportar ya que es fundamental mantener la integridad referencial de nuestra BBDD. Por ejemplo:

- Imaginemos que la tabla `web_direccion` tiene una FK con la tabla `web_paises` por el `id_pais`. En este caso añadir las tablas al `partialDataSet` sería algo como:

```

view plain copy to clipboard print ?
01. ....
02. ....
03.
04.         QueryDataSet partialDataSet = new QueryDataSet(connection);
05.         // Specify the SQL to run to retrieve the data
06.         partialDataSet.addTable("web_paises");
07.         partialDataSet.addTable("web_direccion");
08.         partialDataSet.addTable("web_usuario");
09.
10. ....
11. ....

```

Vemos que el orden de la inserción hace que se mantenga la integridad referencial, ya que añadimos la tabla de `web_pais` antes que la tabla `web_direccion`, por lo que no habría ningún problema a la hora de la importación; si lo hiciésemos al revés, añadiendo al dataset la tabla `web_direccion` antes que la tabla `web_paises`, la importación desde el XML generado fallaría, ya que estaríamos intentando insertar direcciones con países que aún no existen en nuestra BBDD.

Otra ventaja de realizar exportaciones parciales es que tenemos la posibilidad de exportar datos obtenidos a través de una consulta. Por ejemplo:

- Imaginemos que nuestro test solo necesita los usuarios cuyo `id_usuario > 5`.

```

view plain copy to clipboard print ?
01. ....
02. ....
03.
04.         QueryDataSet partialDataSet = new QueryDataSet(connection);
05.         // Specify the SQL to run to retrieve the data
06.         partialDataSet.addTable("web_paises");
07.         partialDataSet.addTable("web_direccion");
08.         partialDataSet.addTable
("web_usuario","select * from web_usuario where id_usuario>5");
09.
10. ....
11. ....

```

6.2 Exportación total

Cuando hablamos de exportación total hemos de aclarar que no significa que no podamos utilizar `QueryDataSet`, es más, como veremos a continuación, trabajar con `QueryDataSet` nos ayudará a mantener la integridad referencial de nuestra BBDD. La diferencia respecto al punto anterior estriba en el número de tablas involucradas en la exportación. En una exportación total de la base de datos es posible que haya un número considerable de tablas; por tanto, la opción anterior no sería recomendable, debido a que mantener la integridad referencial de nuestra BBDD sería una tarea muy costosa.

A continuación se muestra cómo quedaría la implementación de una exportación total:

```

view plain copy to clipboard print ?
01. ....
02. ....
03. public static void generateXML(String driverName,
04.                               String urlDB,
05.                               String userDB,
06.                               String passwordDB,
07.                               String schemaBD,
08.                               String nameXML) throws SQLException {
09.
10.     Connection conn=null;
11.
12.     try {
13.         // Connect to the database
14.         DriverManager.registerDriver((Driver)Class.forName(driverName).newInstance());
15.         conn = DriverManager.getConnection(urlDB, userDB, passwordDB);
16.         IDatabaseConnection connection = new DatabaseConnection(conn, schemaBD);
17.
18.
19.         DatabaseSequenceFilter filter = new DatabaseSequenceFilter(connection);

```

Últimas ofertas de empleo

2009-07-31
T. Información - Operador (día / noche) - BARCELONA.

2009-06-25
Atención a cliente - Call Center - BARCELONA.

2009-06-19
Otras - Ingeniería (minas, puentes y puertos) - VALENCIA.

2009-06-17
Comercial - Ventas - ALICANTE.

2009-06-03
Comercial - Ventas - VIZCAYA.

· Anuncios Google
· [Delphi SQL](#)
· [Componentes Delphi](#)
· [Excel to Access](#)
· [Java MS Access](#)

```

20.     IDataset datasetAll = new FilteredDataSet(filter, connection.createDataSet());
21.     QueryDataSet partialDataSet = new QueryDataSet(connection);
22.
23.     String[] listTableNames = filter.getTableNames(datasetAll);
24.     for (int i = 0; i < listTableNames.length; i++) {
25.         final String tableName = listTableNames[i];
26.         // Specify the SQL to run to retrieve the data
27.         partialDataSet.addTable(tableName);
28.     }
29.
30.
31.     // Specify the location of the flat file(XML)
32.     FlatXmlWriter datasetWriter = new FlatXmlWriter(new FileOutputStream
("C:\\\" + nameXML + ".xml"));
33.
34.     // Export the data
35.     datasetWriter.write(partialDataSet);
36.
37.     } catch (Exception exc) {
38.         exc.printStackTrace();
39.
40.     } finally{
41.         conn.close();
42.     }
43. }
44.
45. ....
46. ....

```

Lo más importante a destacar es cómo DBUnit proporciona una herramienta muy valiosa, como es la clase DatabaseSequenceFilter. A través de su método getTableNames(dataset) podemos recuperar un array con los nombres de las tablas en el orden correspondiente, según las relaciones entre las mismas. Esto supone una gran ventaja, ya que si nuestra base de datos tiene 200 tablas, con DatabaseSequenceFilter recuperamos el orden de inserción adecuado para generar nuestro XML, y sabremos al 100 % que nuestra BBDD mantendrá la integridad referencial.

7. Importando BBDD.

Una opción muy extendida al realizar los test unitarios es crear una base de datos en memoria. En nuestro caso será Hsqldb; por lo tanto, estaríamos exportando desde PostgreSQL e importando los datos en Hsqldb. El proceso de importación es muy sencillo, si el XML generado con anterioridad es correcto. Lo vemos a continuación:

```

view plain copy to clipboard print ?
01. ....
02. ....
03. public static void createData(String driverName,
04.                               String urlDB,
05.                               String userDB,
06.                               String password,
07.                               String nameXML) throws SQLException {
08.
09.     Connection conn = null;
10.     try {
11.         // Connect to the database
12.         DriverManager.registerDriver((Driver)Class.forName(driverName).newInstance());
13.         conn = DriverManager.getConnection(urlDB, userDB, password);
14.         IDatabaseConnection connection = new DatabaseConnection(conn);
15.
16.         DatabaseOperation.INSERT.execute(connection, new FlatXmlDataSet(new FileInputStream
("C:\\\" + nameXML + ".xml")));
17.
18.     } catch (Exception exc) {
19.         exc.printStackTrace();
20.     } finally{
21.         conn.close();
22.     }
23. }
24.
25. ....
26. ....

```

Como podemos ver solo hace falta establecer la conexión con base de datos. A continuación, mediante el componente DatabaseOperation indicamos qué operación queremos realizar; en este caso es la operación insertar. Es importante saber que Hsqldb va creando la BBDD no solo a través de la información del XML, sino también de nuestras clases marcadas como entidades; por lo que habrá que prestar atención a la definición de estas, ya que cualquier variación entre las tablas de BBDD y nuestras clases marcadas como entidades, haría que la importación se fuese al traste.

8. Borrando BBDD

Una vez pasado el test unitario correspondiente, es una muy buena costumbre dejar la BBDD en la situación que estaba antes de comenzar la prueba unitaria. Para ello hemos implementado el siguiente método, que es exactamente igual que el de importación pero esta vez el componente DatabaseOperation nos servirá para borrar la BBDD. Vemos la implementación:

```

view plain copy to clipboard print ?
01. ....
02. ....
03. public static void deleteData(String driverName,
04.                               String urlDB,
05.                               String userDB,
06.                               String password,
07.                               String nameXML) throws SQLException {
08.
09.     Connection conn = null;
10.     try {
11.         // Connect to the database
12.         DriverManager.registerDriver((Driver)Class.forName(driverName).newInstance());
13.         conn = DriverManager.getConnection(urlDB, userDB, password);
14.         IDatabaseConnection connection = new DatabaseConnection(conn);
15.
16.         DatabaseOperation.DELETE.execute(connection, new FlatXmlDataSet(new FileInputStream
("C:\\\" + nameXML + ".xml")));
17.
18.     } catch (Exception exc) {
19.         exc.printStackTrace();
20.     } finally{
21.         conn.close();
22.     }
23. }
24.
25. ....
26. ....

```

```

20.     }
21.   }
22.   .....
23.   .....

```

9. Test JUnit

Implementación de test unitario con JUnit:

```

view plain copy to clipboard print ?
01. package com.autentia.dbunit.test;
02.
03. import java.util.List;
04.
05. import javax.annotation.Resource;
06.
07. import org.junit.AfterClass;
08. import org.junit.Assert;
09. import org.junit.BeforeClass;
10. import org.junit.Test;
11. import org.junit.runner.RunWith;
12. import org.springframework.test.context.ContextConfiguration;
13. import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
14. import org.springframework.test.context.transaction.TransactionConfiguration;
15. import org.springframework.transaction.annotation.Transactional;
16.
17. import com.autentia.dbunit.persistence.entity.WebUsuarios;
18. import com.autentia.dbunit.persistence.entity.WebDirecciones;
19. import com.autentia.dbunit.persistence.entity.WebPaises;
20. import com.autentia.dbunit.persistence.Dao;
21.
22.
23. @RunWith(SpringJUnit4ClassRunner.class)
24. @ContextConfiguration(locations = { "classpath:applicationDBUnitContext-test.xml" })
25. @TransactionConfiguration(transactionManager = "transactionManager", defaultRollback = false)
26.
27. @Transactional
28. public class DbUnitTest {
29.
30.     @Resource
31.     private Dao dao;
32.
33.     @BeforeClass
34.     public static void setUp() {
35.
36.         try {
37.             DBUnitUtils.generateXML("org.postgresql.Driver",
38.                                     "jdbc:postgresql://localhost:5432/dbunit",
39.                                     "postgres",
40.                                     "postgres",
41.                                     "public",
42.                                     "dbUnit");
43.             DBUnitUtils.createData("org.hsqldb.jdbcDriver",
44.                                    "jdbc:hsqldb:file:/tmp/appName/db/hsqldb/hibernate/shutd
45.
46.                                    "sa",
47.                                    "",
48.                                    "dbUnit");
49.
50.         } catch (Exception e) {
51.             // TODO Auto-generated catch block
52.             e.printStackTrace();
53.         }
54.
55.     @AfterClass
56.     public static void tearDown() {
57.
58.         try {
59.             DBUnitUtils.deleteData("org.hsqldb.jdbcDriver",
60.                                    "jdbc:hsqldb:file:/tmp/appName/db/hsqldb/hibernate/shutd
61.
62.                                    "sa",
63.                                    "",
64.                                    "dbUnit");
65.
66.         } catch (Exception e) {
67.             // TODO Auto-generated catch block
68.             e.printStackTrace();
69.         }
70.
71.     @Test
72.     public void test1() {
73.
74.         // Comprueba la tabla de paises contenga datos
75.         List<WebPaises> paises = dao.loadAll(WebPaises.class);
76.         Assert.assertTrue(paises != null);
77.         Assert.assertTrue(paises.size() > 0);
78.
79.         // Comprueba la tabla de direcciones contenga datos
80.         List<WebDirecciones> direcciones = dao.loadAll(WebDirecciones.class);
81.         Assert.assertTrue(direcciones != null);
82.         Assert.assertTrue(direcciones.size() > 0);
83.
84.         // Comprueba la tabla de usuarios contenga datos
85.         List<WebUsuarios> usuarios = dao.loadAll(WebUsuarios.class);
86.         Assert.assertTrue(usuarios != null);
87.         Assert.assertTrue(usuarios.size() > 0);
88.
89.
90.     }
91. }

```

Como vemos, este test podría ser como cualquiera de los que ya hayamos realizado en alguna otra ocasión. Lo más importante a destacar son los métodos `startUp` y `tearDown` anotados con `@BeforeClass` y `@AfterClass` respectivamente, lo que quiere decir que se ejecutarán antes y después del test. Por tanto, antes de lanzar el test, generamos el XML y creamos la BBDD, importando los datos del XML generado. Después del test borramos la BBDD dejándola en el estado inicial.

10. Conclusiones.

Siempre que nuestras pruebas unitarias necesiten un conjunto de datos con el que interactuar, DBUnit es una eficaz herramienta para llevar a cabo esta misión. Sin embargo, no siempre es tan sencillo obtener un juego de datos adecuado, ni siquiera con DBUnit, por lo que podemos decir que ha esta librería todavía le queda mucho por hacer en este sentido. Para finalizar, me gustaría destacar una característica más sobre DBUnit, y es que podemos hacer test propios de BBDD, es decir, podemos comparar tablas, columnas, tipo de datos etc entre distintas BBDD. Sobre esta cuestión profundizaremos en próximos tutoriales.

Un saludo.

Saul

<mailto:sgdiaz@autentia.com>

¿Qué te ha parecido el tutorial? Déjanos saber tu opinión y ivota!

Muy malo Malo Regular Bueno Muy bueno



Votar

Anímate y coméntanos lo que pienses sobre este tutorial

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Nombre: E-Mail:

Comentario:

Enviar comentario

[Texto Legal y condiciones de uso](#)

- Puedes inscribirte en nuestro servicio de notificaciones [haciendo clic aquí](#).
- Puedes firmar en nuestro libro de visitas [haciendo clic aquí](#).
- Puedes asociarte al grupo [AdictosAlTrabajo](#) en XING [haciendo clic aquí](#).
- Añadir a favoritos Technorati. 



Esta obra está licenciada bajo [licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

Recuerda

Autentia te regala la mayoría del conocimiento aquí compartido ([Ver todos los tutoriales](#)). Somos expertos en: J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ... y muchas otras cosas.

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?, ¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos ...

Autentia = Soporte a Desarrollo & Formación.

info@autentia.com

J2EE, EJBs, Struts...

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf
Selenium IDE- Incorporando while en los test	En este tutorial veremos como ampliar los comandos disponibles en Selenium IDE adaptandolos a nuestras necesidades.	2009-08-20	10	-	-	
Joomla 1.5. Instalación y configuración	Veamos en detalle cómo instalar Joomla 1.5 y aplicar algunas configuraciones posteriores de las disponibles en este CMS	2009-08-14	386	Bueno	5	
Hibernate Search, Bridges, Analizadores y más	Detalles del uso de Hibernate Search con bases de datos Lucene	2009-07-29	397	Muy bueno	2	
AOP con AspectJ y Maven	Programacion orientada a aspectos con AspectJ y Maven	2009-07-08	793	Bueno	1	
JMeter, Pruebas de stress sobre aplicaciones web: Grabando y reproduciendo navegaciones	En este tutorial Carlos García nos enseñará a grabar y reproducir navegaciones con JMeter, para poder realizar pruebas de carga o stress sobre aplicaciones Web	2009-07-05	1657	Bueno	9	
Configuración de la desconexión de usuarios con ICEFaces	Este tutorial muestra la manera de configurar y traducir la ventana de desconexión o pérdida de sesión del usuario en ICEFaces.	2009-06-15	1573	Muy bueno	10	
Plugin Hibernate3 para Maven	En este tutorial veremos las posibilidades que nos ofrece el plugin de Hibernate3 para Maven, como por ejemplo, la generación del esquema de base de datos desde clases con anotaciones.	2009-05-02	1355	Bueno	8	
Informes Crosstab con iReport	Con este tutorial vamos a ampliar el nivel de conocimiento sobre iReport enseñando como hacer un informe usando crosstab o tablas dinámicas	2009-04-16	4050	Bueno	25	
Tests de Selenium con librerías de componentes JSF: Apache Tomahawk.	En este tutorial vamos a hablar de cómo escribir tests funcionales con Selenium IDE sobre interfaces de usuario construidas con librerías de componentes visuales JSF y, en concreto, con Apache Tomahawk y uno de sus componentes.	2009-04-14	2008	Muy bueno	2	
Registro de Web Services con Apache Juddi. Configuración y ejemplo	Veamos como podemos catalogar y buscar web services bajo la especificación UDDI (Universal Description, Discovery and Integration)	2009-04-13	2641	Muy bueno	14	

Nota:

Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento. Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores. En algún caso se puede hacer referencia a marcas o nombres cuya propiedad y derechos es de sus respectivos dueños. Si algún afectado desea que incorporemos alguna reseña específica, no tiene más que solicitarlo. Si alguien encuentra algún problema con la información publicada en este Web, rogamos que informe al administrador rcanales@adictosaltrabajo.com para su resolución.

· [Anuncios Google](#) · [Export Excel to XML](#) · [Delphi 7 Components](#) · [Ado SQL](#) · [Managed Data](#) · [C# UDB](#)

Copyright 2003-2009 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

