Lenguaje Java Avanzado

Sesión 8: Pruebas con DbUnit

Puntos a tratar

- Framework
- Prácticas recomendadas
- Ciclo de vida
- Interfaces y clases
- DbUnit y Eclipse
- Ejemplos

Pruebas con la base de datos

- Necesidad de probar que la capa de acceso a datos accede correctamente a la base de datos.
- Necesidad de probar el código dados determinados estados de la base de datos.

DbUnit

- Framework de código abierto creado por Manuel Laflamme
- Basado en JUnit, es una extensión
- Permite gestionar el estado de una base de datos durante las pruebas unitarias y e integración.
- Alternativa a la creación de stubs y mocks para controlar las dependencias externas.
- Se puede usar en conjunción con JUnit.

Ciclo de vida con DbUnit

- Ciclo de vida
  1. Eliminar el estado previo de la BD resultante de pruebas anteriores (en lugar de restaurarla tras cada test)
  2. Cargar los datos necesarios para las pruebas de la BD (sólo los necesarios para cada test)
  3. Utilizar los métodos de la librería DbUnit en las aserciones para realizar el test

Prácticas recomendadas en DbUnit

- Usar una instancia de la BD por cada desarrollador.
- Programar los tests para no tener que limpiar la base de datos después.
- Usar múltiples conjuntos de datos pequeños en lugar de uno grande.
- Inicializar los datos comunes sólo una vez para todos los tests.
- Estrategias de conexión
  - Cliente remoto con DatabaseTestCase
  - A través del pool de conexiones del servidor con IdataBaseConnection o con JndiBasedDBTestCase

Características de DbUnit

- Permite evitar los problemas que surgen si el último caso de prueba ha dejado la base de datos inconsistente.
- Puede trabajar con conjuntos de datos grandes
- Permite verificar que el contenido de la base de datos es igual a determinado conjunto de datos esperado, a nivel de fichero, de consulta y de tabla.
- Mecanismo basado en XML para cargar y exportar los datos.
- Casos de prueba individuales por cada operación.

Clases e interfaces de DbUnit

- DBTestCase
  - Hereda de JUnit TestCase
  - Proporciona métodos para inicializar y restaurar la BD antes y después de cada test
  - Utiliza IDatabaseTester para conectar con la BD
- Alternativa: Utilizar directamente IDatabaseTester (desde la versión 2.2 de DbUnit). Utilizaremos la versión 2.4.8.

Clases e interfaces de DbUnit

- IDataBaseTester devuelve conexiones a la BD, de tipo IDatabaseConnection
  - Implementaciones: JdbcDatabaseTester entre otras
  - Métodos:
    - OnSetUp( ), setSetUpOperation(op)
    - OnTearDown( ), setTearDownOperation(op)
    - getConnection( )

Clases e interfaces de DbUnit

- IDataBaseConnection
- Métodos:
  - createDataSet( ) - crea un dataset para la BD entera
  - createDataSet(lista de tablas)
  - createTable(tabla) - crea una tabla extraída de la BD
  - createQueryTable(tabla, sql) - crea una tabla con el resultado de la query sql sobre la BD
  - getConnection( ), getConfig( ), getRow( )

Clases e interfaces de DbUnit

- IDataSet: Representa una colección de tablas. Se utiliza para situar la BD en un estado determinado y para comparar el estado actual de la BD con el estado esperado
- Implementaciones:
  - FlatXmlDataSet - lee/escribe datos en formato xml
  - QueryDataSet - guarda colecciones de datos resultantes de una query
- Métodos:
  - getTable(tabla)- devuelve la tabla especificada

Clases e interfaces de DbUnit

- FlatXMLDataSet: importa y exporta conjuntos de datos a XML

```
<!DOCTYPE dataset SYSTEM "my-dataset.dtd">
<dataset>
   <TEST_TABLE COL0="row 0 col 0"
      COL1="row 0 col 1"
      COL2="row 0 col 2"/>
   <TEST_TABLE COL1="row 1 col 1"/>
   <SECOND_TABLE COL0="row 0 col 0"
      COL1="row 0 col 1" />
   <EMPTY_TABLE/>
</dataset>
```

Clases e interfaces de DbUnit

- Clase Assertion: define los métodos estáticos para realizar las comparaciones:
  - assertEquals(IDataSet, IDataSet)
  - assertEquals(ITable, ITable)

Dependencias de librerías

- JUnit (utilizaremos la versión 4)
- Jakarta Commons IO (versión 1.4)
  - Utilidades para E/S
- Slf4j (versión 1.6)
  - Frontend para frameworks de logging

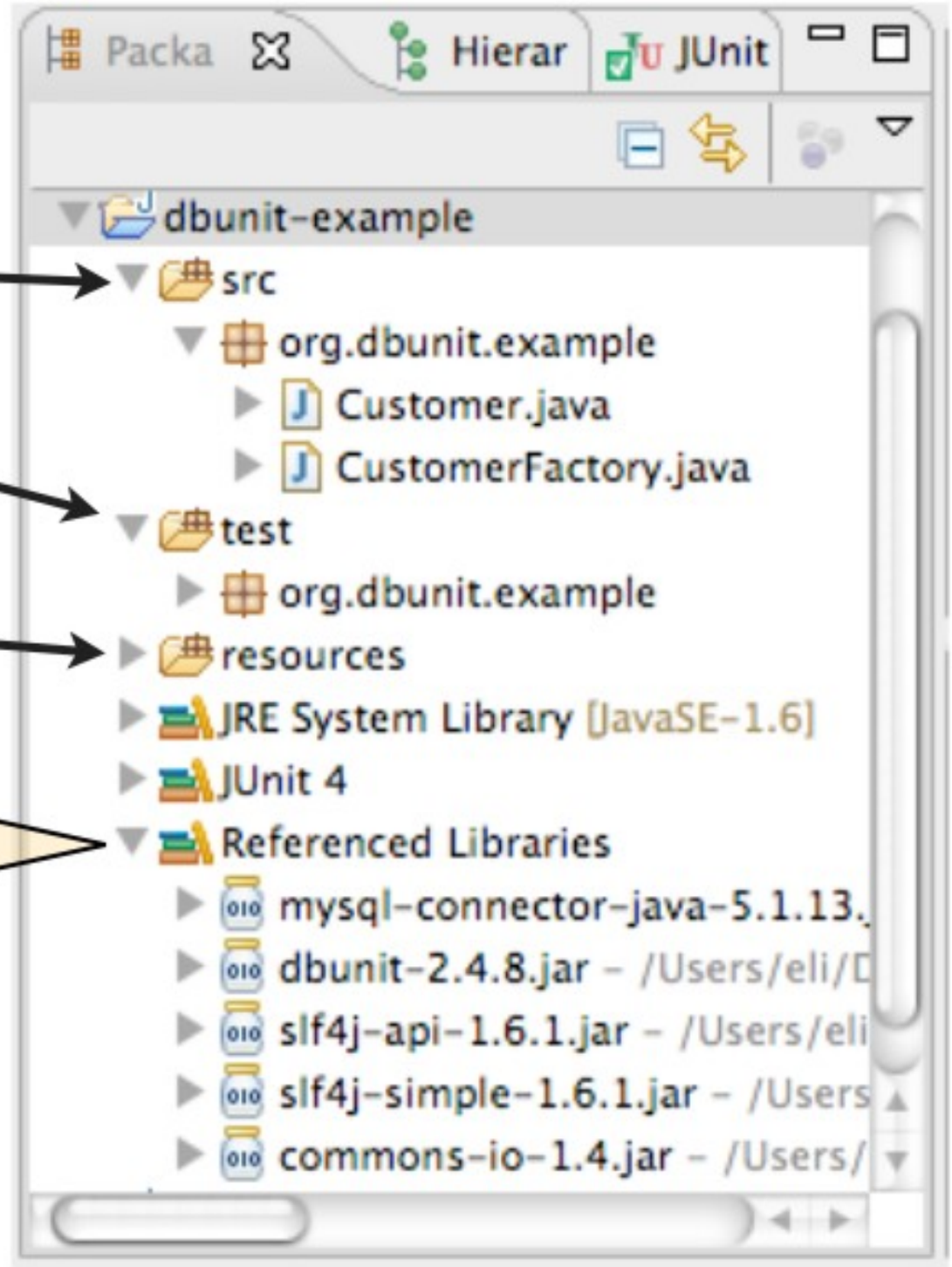
En Eclipse

Fuentes

Fuentes pruebas

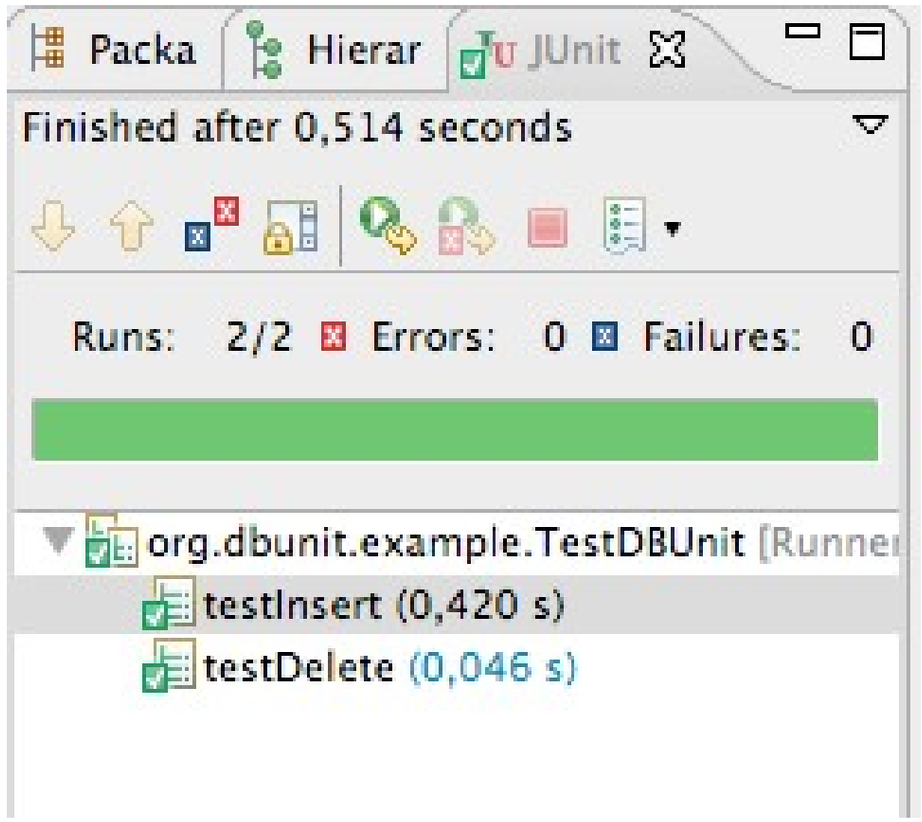
Recursos

mysql-connector-5.1.13
dbunit-2.4.8
slf4-api-1.6.1
slf4-simple-1.6.1
commons-io-1.4



En Eclipse

- Ejecutamos la prueba con Run as → JUnit Test



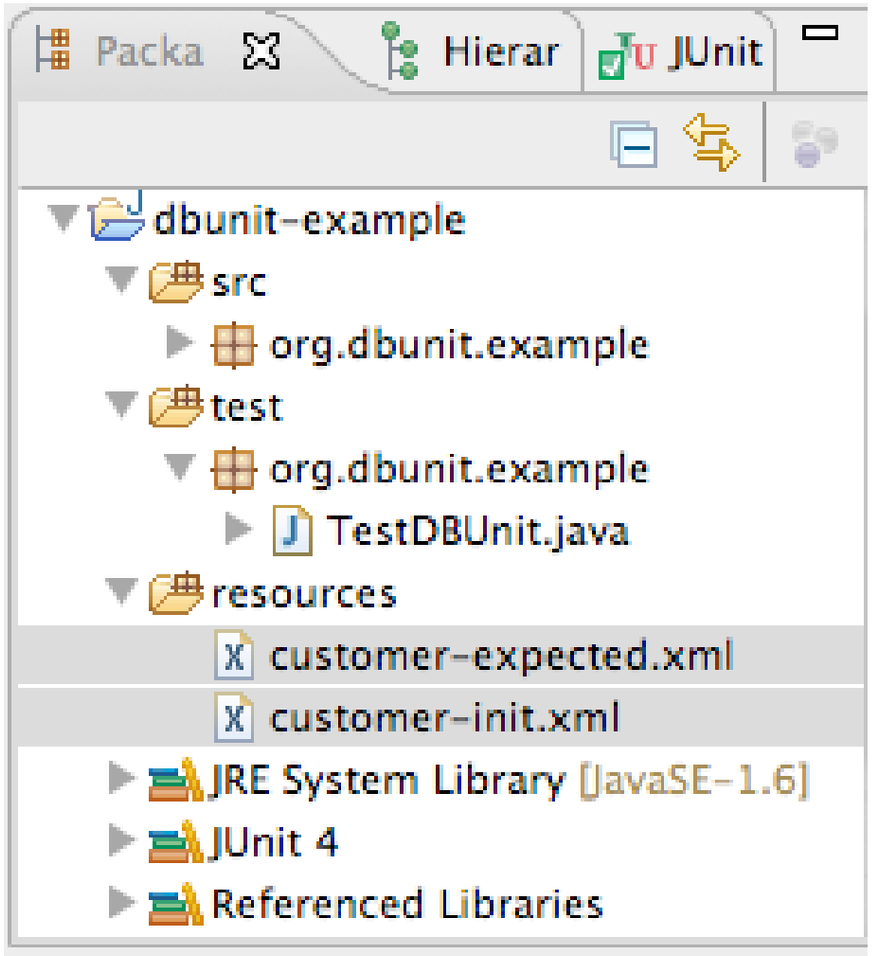
Estructura de la prueba

```
public class TestMiJDBCDAO {
    private JDBCDAO dao;
    private IDatabaseTester databaseTester;

    @Before
    public void setUp() throws Exception {



    }

    @After
    public void tearDown() throws Exception {

    }


    @Test
    public void test1() throws Exception {


    }
    ...
}
```

Antes y después de cada test

```
@Before
public void setUp() throws Exception {
      //Obtener instancia del DAO que testeamos
      pdao = new JDBCPeliculaDAO();

      //Acceder a la base de datos
      databaseTester = new JdbcDatabaseTester("com.mysql.jdbc.Driver",
            "jdbc:mysql://localhost/databasename", "username", "password");

      //Inicializar el dataset en la BD
       FlatXmlDataSetBuilder builder = new FlatXmlDataSetBuilder();
      IDataSet dataSet = builder.build(
                        this.getClass().getResourceAsStream("/db-init.xml"));
      databaseTester.setDataSet(dataSet);

          //Llamar a la operación por defecto setUpOperation
          databaseTester.onSetup();
}


@After
public void tearDown() throws Exception {
      databaseTester.onTearDown();
}
```

Recursos XML



customer-init.xml | customer-expected.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <dataset>
3
4     <customer />
5 4
6 </dataset>
```

customer-init.xml | customer-expected.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <dataset>
3     <customer id="1"
4               firstname="John"
5               lastname="Smith"
6               street="1 Main Street"
7               city="Anycity" />
8 </dataset>
```

Prueba

```
@Test
public void test1DelAdd() throws Exception {
      //Realizar la operación con el JDBC DAO
      ...

      //Conectar a la base de datos MySQL
      IDatabaseConnection connection = databaseTester.getConnection();
      DatabaseConfig dbconfig = connection.getConfig();
      dbconfig.setProperty("http://www.dbunit.org/properties/datatypeFactory", new MySqlDataTypeFactory());

      //Crear el DataSet a partir de la base de datos MySQL
      QueryDataSet partialDataSet = new QueryDataSet(connection);
      partialDataSet.addTable("tabla1");
      partialDataSet.addTable("tabla2");
      partialDataSet.addTable("tabla3");

      //Escribir el DataSet en XML para después compararlo con el esperado
      File outputFile = new File("db-result.xml");
      FlatXmlDataSet.write(partialDataSet, new FileOutputStream(outputFile));

      //Obtener los datos esperados del XML
      URL url = IDatabaseTester.class.getResource("/db-expected.xml");
      Assert.assertNotNull(url);
      File inputFile = new File(url.getPath());

      //Comparar los ficheros XML
      Assert.assertEquals(FileUtils.readFileToString(inputFile, "UTF8"),
                          FileUtils.readFileToString(outputFile, "UTF8"));

}
```

¿Preguntas...?