PERSISTENCIA Y ACCESO A BASE DE DATOS. HERRAMIENTAS

Práctica 4 . Anexo

Prácticas Ingeniería del Software
ETS Ingeniería Informática
DSIC – UPV

Curso 2014-2015

Contenidos

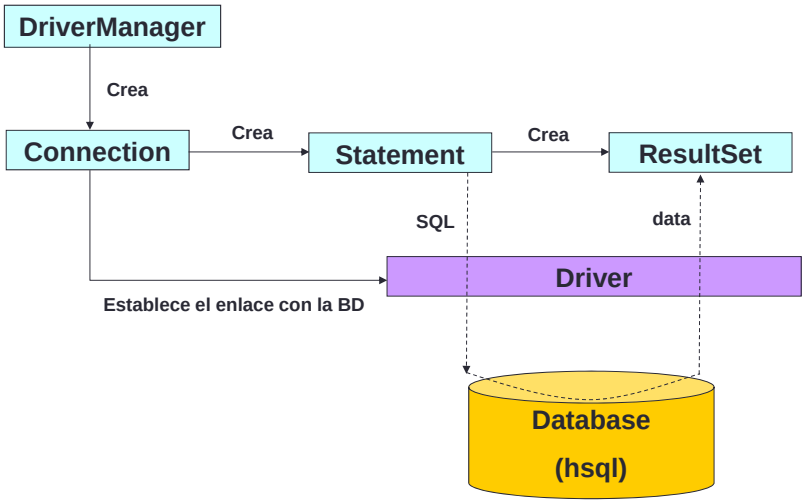
- ✓ Introducción a JDBC. Funcionamiento básico. Consultas y actualizaciones en SQL
- ✓ Base de Datos HSQLDB
- ✓ Herramientas en Eclipse
  - ✓ Clay
  - ✓ SQLExplorer

JDBC

Java Data Base Connectivity (JDBC):

- ✓ Biblioteca que proporciona los medios necesarios para efectuar consultas SQL y acceder y operar con una base de datos relacional.
- ✓ Se diseñó como una interfaz de programación de aplicaciones (API) orientada a objetos para acceder a bases de datos.
- ✓ Implementada en el paquete **java.sql**.

Introducción: Cómo funciona JDBC



Ejemplo

```
import java.sql.*;
public class Ejemplo {
  public static void main(String[] args) {
   try {
   Class.forName("org.hsqldb.jdbcDriver");

   String sourceURL = "jdbc:hsqldb:hsql://localhost/videoteca";
   Connection dbcon = DriverManager.getConnection(sourceURL);
   Statement sentencia = dbcon.createStatement();
   String ins = "INSERT INTO GENERO VALUES(null,'TERROR')";
   sentencia.executeUpdate(ins);
   ResultSet resultado = sentencia.executeQuery("select * from genero");
   // Mostrar los datos
   while (resultado.next()) {
   System.out.println(resultado.getInt("ID") + "  " + resultado.getString("NOMBRE"));
   }
   dbcon.close();
  } catch (ClassNotFoundException cnf) {
                  System.out.println("Driver erróneo " + cnf);
  } catch (SQLException sqle) {
                  System.out.println("Error de SQL " + sqle);
  }
  }
}
```

Introducción: Cargar el driver JDBC

- Para cargar el driver se llamará al método **forName()** de la clase **Class**, pasándole como argumento el tipo de driver.
- Puede lanzar una excepción del tipo **ClassNotFoundException** si no puede encontrar la clase del driver.

```
import java.sql.*;
try {
  ...
  Class.forName ("org.hsqldb.jdbcDriver")
  ...
}
catch (ClassNotFoundException ex) {

// Incluir el jar hsqldb.jar
// Project | Properties | Add External Jars…
    …
}
```

Conexión con una base de datos

- Se utiliza el método **getConnection**() de la clase **DriverManager**.
- El argumento es un objeto String que define el *URL (Uniform Resource Locator)* donde está la base de datos.
- Los URLs de JDBC tienen el siguiente formato:

jdbc:<subprotocolo>:<identificador de origen de datos>

donde,

- <subprotocolo>: identifica el driver JDBC a utilizar, para HSQLDB el valor **hsqldb**.
- <identificador de origen de datos>: depende del driver en HSQLDB es el nombre de la BD.

```
String fuenteURL = "jdbc:hsqldb:hsql://localhost/nombre_base_datos"
Connection DBConnection = DriverManager.getConnection(fuenteURL);
```

Consultas y actualizaciones en SQL

- Interfaz **Statement**
  - La interfaz Statement proporciona métodos para ejecutar sentencias SQL y obtener los resultados.
  - Se obtiene llamando al método createStatement() de un objeto Connection válido:

```
Statement sentencia = connection.createStatement();
```

  - Métodos:
    - executeQuery(): para sentencias SQL que recuperen datos (SELECT). Devuelve un objeto ResultSet con la tabla generada por la consulta.
    - executeUpdate(): para realizar actualizaciones. Sentencias DML (INSERT,DELETE,UPDATE) o DDL(CREATE TABLE,ALTER TABLE, DROP TABLE). Devuelve un entero con las filas que se vieron afectadas, en el caso de DDL devuelve 0.
    - execute(). Para ejecutar sentencias que devuelven más de un ResultSet, más de un valor de actualizaciones o una combinación de ambos.

Consultas SQL: executeQuery() y ResultSet

- El método executeQuery() devuelve un objeto que implementa el interfaz ResultSet, y que contiene la tabla generada por la consulta.
- El ResultSet contiene un cursor o puntero que puede manipularse para acceder a cualquier fila de la tabla resultado. Inicialmente apunta a una posición anterior a la primera fila.
- Métodos:
  - next() desplaza el cursor a la próxima posición. Devuelve el valor false si el movimiento desplaza a una fila fuera de la tabla.
  - getXXX(String nombrecolumna) o getXXX(int indicecolumna): acceden al valor de tipo XXX de cualquier columna de la fila actual. Índice es la posición de la columna en la SELECT, empezando por 1.

```
getInt("ID") o getInt(1)
getString("NOMBRE") o getString(2)
```

Cursores dinámicos

- La interfaz ResultSet aporta otros métodos para navegar:
  - previous(): se mueve a la fila anterior
  - first(): se mueve a la primera fila
  - last(): se mueve a la última fila
  - absolute(int n): se mueve a la fila enésima del ResultSet
  - relative(int n): se mueve n filas desde la fila en la que está
  - updateXXX(String columna, XXX valor), o
  - updateXXX(int indice, XXX valor): modifica la columna con el valor especificado
  - updateRow(): cambia en la BD con los contenidos de la fila actual
  - moveToInsertRow(): se mueve a una fila especial que se utiliza para insertar una fila de datos nueva en la tabla. Si se mueve el cursor a una fila distinta antes de llamar a insertRow() la BD no se actualizará
  - insertRow(): inserta en la base de datos una nueva fila
  - moveToCurrentRow(): retrasa el cursor a la posición que estaba antes de que el método moveToInsertRow() fuese llamado
  - cancelRowUpdates(): cancela los cambios hechos a la fila actual
  - deleteRow(): Borra la fila actual del ResultSet y de la base de datos

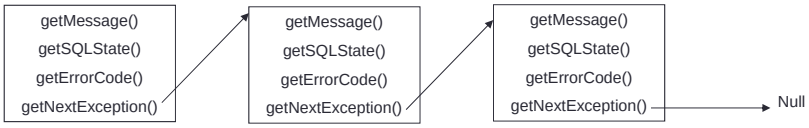
Gestión de errores

- Se manejan con la excepción **`SQLException`**.
- Tiene tres datos importantes:
  - **El mensaje de la excepción**. Lo devuelve el método getMessage(). Varía según el driver JDBC utilizado.
  - **El estado SQL**. Devuelto con el método getSQLState(). Cadena que contiene un estado definido en el estándar X/Open SQL.
  - **Código de error del fabricante**. Valor entero devuelto con el método getErrorCode(). Su significado depende por completo del fabricante del driver.

```
catch (SQLException sqle) {
    System.out.println("Texto " + sqle.getMessage());
    System.out.println("State " + sqle.getSQLState());
    System.out.println("Vendedor " + sqle.getErrorCode());
 }
```


Encadenamiento de Excepciones

- Cuando se lanza una excepción de SQL, es posible que haya más de un objeto excepción asociado con el error. Podemos enlazar las excepciones con el método **getNextException().**




```
catch (SQLException sqle) {
      do {
        System.out.println("Texto " + sqle.getMessage());
        System.out.println("State " + sqle.getSQLState());
        System.out.println("Vendedor " + sqle.getErrorCode());
      }
      while ( (sqle = sqle.getNextException()) != null);
    }
```

HERRAMIENTAS

-Base de Datos: HSQLDB

- Eclipse SQLExplorer

- Clay

HSQLDB

http://www.**hsqldb**.org

- **HSQLDB (Hypersonic SQL Data Base)** es un completo gestor de bases de datos relacionales 100% puro Java y de código abierto.

- Características:
  - Tiempo de arranque mínimo y gran velocidad en las operaciones: SELECT, INSERT, DELETE y UPDATE
  - **Sintaxis SQL estándar ,** Integridad referencial (claves foráneas) , Procedimientos almacenados en Java
  - Triggers.
  - Utilizado por los programadores para desarrollar y testear.
  - Muy estable.
  - Fácil de instalar: **Se instala descomprimiendo en un directorio del disco local.**
  - **Sólo se necesita un. jar para ejecutarse.**

- Compuesto de varios directorios:
  - **lib**. Se encuentra el **hsqldb.jar.**
  - **demo.** Está ubicado un bat llamado **runServer** para el arranque en modo servidor. (o en **Bin, según la versión instalada).**
  - **data**. Ubicación de las BD disponibles. Al arrancar una base de datos que no existe se crea una nueva vacía en el directorio data.
  - ....

HSQLDB

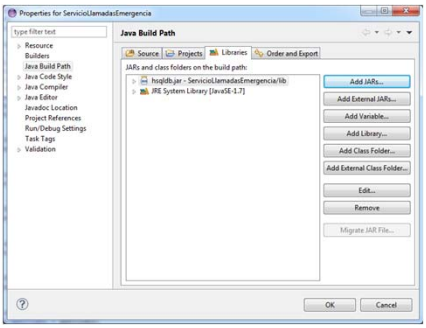
- Cada BD de HSQLDB consiste de entre 2 a 5 ficheros. Todos se llaman igual que la BD pero con distinta extensión, situados en el mismo directorio.
  - **.properties*. Configuración general de la BD.
  - **.scripts*. Definiciones de tablas y otros objetos, además de los datos para un tipo de tabla no cacheada.
  - **.log*. Registra los cambios hechos en los datos. Se borra cuando se cierra correctamente la BD.
  - **.data*. Se incluyen los datos de las tablas cacheadas.
  - **.backup*. Copia estable de la BD.
- Por cuestiones de integridad se recomienda copiar los ficheros con la BD cerrada y todos juntos.
- Se puede arrancar de 3 modos:

➡ *Servidor*. Se ejecuta en una JVM diferente. Es el aconsejado.

  - *Standalone*. El motor de BD se ejecuta como parte del programa aplicación.
  - *Memory-only*. No es persistente y los datos existen exclusivamente en memoria.

HSQLDB

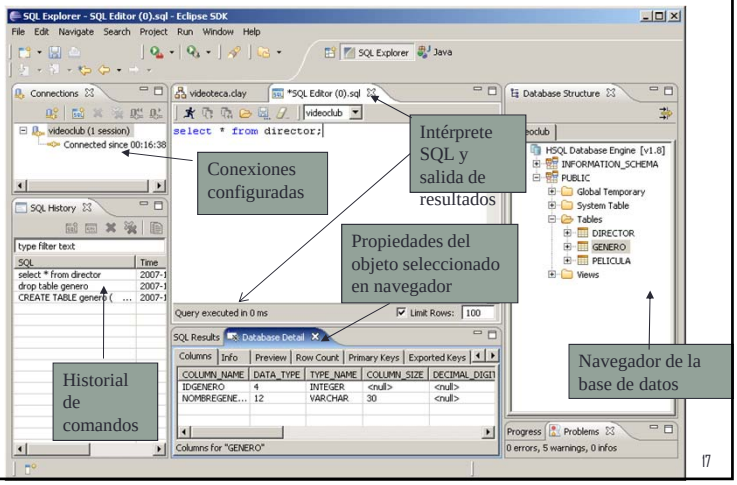
- **Para acceder a la BD desde un proyecto eclipse** tenemos que incluir el **hsqldb.jar** como librería
- Se hace desde propiedades del proyecto / Java Build Path / Libraries / Add JARs...



Eclipse SQLExplorer

- Cliente ligero de SQL que permite visualizar y consultar tablas u objetos de cualquier Base de Datos compatible con JDBC.
- Se debe configurar en **Window/Preferences/SQL Explorer/JDBC Drivers**, el driver JDBC de HSQLDB apuntando a donde se encuentre hsqldb.jar en la pestaña **Extra Class Path**.

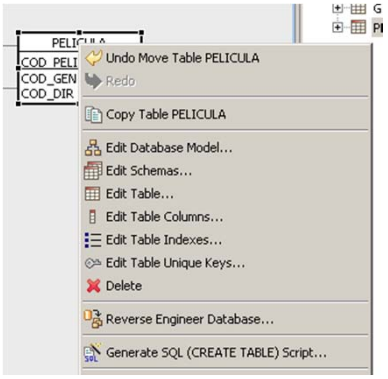
- Se accede desde la **perspectiva SQLExplorer**



Eclipse Clay

- Plugin que permite **representar el modelo relacional de una base de datos**. Podemos crear el modelo partiendo de cero o bien por ingeniería inversa de una base de datos existente

- El modelo permite representar:
  - tablas, columnas y sus características
  - claves ajenas entre tablas
  - claves únicas e índices



- **Genera un script SQL** para la creación de la base de datos. El script se utiliza en SQL Explorer para crear la base de datos.