

TRABAJO CON BASES DE DATOS MEDIANTE JDBC

ADOLFO SANZ DE DIEGO

JUNIO DE 2011

1 LA API JDBC

- **Java DataBase Connectivity** es una API que permite la ejecución de operaciones sobre casi cualquier bases de datos.
- Es una **especificación** (una serie de Interfaces).
- Para utilizar una base de datos en particular, el usuario necesita tener los **drivers** (la implementación) específicos de esa base de datos.
- Una vez establecida una conexión con la base de datos el usuario puede realizar cualquier tipo de tarea con la base de datos a las que tenga permiso:
 - **consultas** (SELECT),
 - **actualizaciones** (INSERT, UPDATES),
 - **creación, modificación y borrado de tablas,**
 - **ejecución de procedimientos almacenados,**
 - **etc.**

2 CREAR UNA CONEXIÓN

```
// Parámetros de conexión
String driver    = "com.mysql.jdbc.Driver";
String url       = "jdbc:mysql://localhost/ejemplo";
String user      = "root";
String password  = "Pa$.w0rd";

// Registramos la clase con la que nos vamos a conectar (depende de la B
Class.forName(driver);

// Establecemos una conexión con la base de datos
conexion = DriverManager.getConnection(url, user, password);

// Si no queremos que se ejecute el commit automáticamente
conexion.setAutoCommit(false);
```

3 CERRAR UNA CONEXIÓN

- Antes de cerrar, si queremos hacer un **commit**:

```
conexion.commit();
```

- Antes de cerrar, si queremos hacer un **rollback**:

```
conexion.rollback();
```

- Cerrar siempre en una clausula **finally**

```
conexion.close();
```

4 ACTUALIZACIONES

```
Statement sentencia = null;
try {
    sentencia = conexion.createStatement();
    int numeroDeFilasActualizadas = sentencia.executeUpdate(
        "INSERT... o UPDATE... o DELETE...");
} catch (Exception e) {
    throw e;
} finally {
    try {
        if (sentencia != null) sentencia.close();
    } catch (Exception e) {
        log.warn("error cerrando sentencia", e);
    }
}
```

5 ACTUALIZACIONES CON PARÁMETROS

```
PreparedStatement sentencia = null;
try {
    sentencia = conexion.prepareStatement(
        "INSERT... o UPDATE... o DELETE..." +
        "WHERE COLUMNA1 = ? AND COLUMNA2 = ?");
    sentencia.setInt(1, numeroParaColumna1);
    sentencia.setString(2, cadenaParaColumna2);
    int numeroDeFilasActualizadas = sentencia.executeUpdate();
} catch (Exception e) {
    throw e;
} finally {
    try {
        if (sentencia != null) sentencia.close();
    } catch (Exception e) {
        log.warn("error cerrando sentencia", e);
    }
}
```

6 CONSULTAS

```
Statement sentencia = null;
ResultSet cursor = null
try {
    sentencia = conexion.createStatement();
    cursor = sentencia.executeQuery("SELECT...");
    while(cursor.next()) {
        String fila = cursor.getString("nombreColumna1") + ", "
            + cursor.getString("nombreColumna2") ...;
        System.out.println(fila);
    }
} catch (Exception e) {
    throw e;
} finally {
    // cerrar cursor
    // cerrar consulta
}
```


7 CONSULTAS CON PARÁMETROS

```
PreparedStatement sentencia = null;
ResultSet cursor = null
try {
    sentencia = conexion.prepareStatement(
        "SELECT... WHERE COLUMNA1 = ? AND COLUMNA2 = ?");
    sentencia.setInt(1, numeroParaColumna1);
    sentencia.setString(2, cadenaParaColumna2);
    cursor = sentencia.executeQuery();
    while(cursor.next()) {
        String fila = cursor.getString("nombreColumna1") + ", "
            + cursor.getString("nombreColumna2") ...;
        System.out.println(fila);
    }
} catch (Exception e) {
    e.printStackTrace();
} finally {
    // cerrar cursor
    // cerrar consulta
```

8 IMPRIMIR CONSULTA CON METADATA

```
ResultSetMetaData rsmd = resultSet.getMetaData();
while(cursor.next()) {
    StringBuilder fila = new StringBuilder("");
    for (int i = 1; i <= rsmd.getColumnCount(); i++) {
        fila.append(rsmd.getColumnName());
        fila.append("=");
        fila.append(resultSet.getString(i));
        fila.append("\t");
    }
    System.out.println(fila);
}
```