Especificaciones del driver JDBC

BASE100

BASE 100, S.A.
www.base100.com

Índice

1. Introducción

El driver JDBC del CTSQL cumple con la API 2.0 del JDBC. Se trata de un driver de tipo 4, lo que significa que está implementado íntegramente en Java y que, por lo tanto, puede ser utilizado en cualquier plataforma que soporte dicho lenguaje.

Gracias a este driver podremos acceder directamente a servidores CTSQL en cliente-servidor desde programas escritos en Java.

1.1 Sintaxis de la URL

```
jdbc:ctsql://<dbhost>:<port>/<dbname>[;<attribute-name>=<attribute-value>]*
```

Donde:

<dbhost> Nombre o dirección IP de la Esta es la sintaxis para las URL's del driver JDBC del CTSQL:máquina donde está instalado el servidor CTSQL.

<port> Número del puerto donde espera las conexiones el servidor CTSQL. Este valor debe ser numérico, no siendo válido poner el nombre del servicio.

<dbname> Nombre de la base de datos con la que se quiere establecer conexión.

<attribute-name>=<attribute-value>
Dentro de la URL se pueden indicar las variables de entorno de la conexión separadas por un punto y coma (;). En este apartado será obligatorio indicar al menos el DBPATH. Este formato también permite indicar el usuario y la *password* mediante los <attribute-name> "user" y "password" respectivamente. Los <attribute-name> son sensibles al uso de mayúsculas *(case-sentitive)*.

1.2 Registro del driver

La clase que implementa la interfaz java.sql.Driver, y que por tanto hay que registrar, es la clase **com.transtools.jdbc.CtsqlJdbcDriver.** La API JDBC 1.0 permite dos formas para registrar el driver:

- Cargando explícitamente el driver utilizando el método Class.forName, de la siguiente forma:

```
Class.forName("com.transtools.jdbc.CtsqlJdbcDriver");
```

- Cuando la clase java.sqlDriverManager del JDBC arranca busca el nombre del driver en la propiedad "sql.drivers" de las propiedades del sistema. Si existe dicha propiedad debe indicar una lista de drivers separados por un punto y coma (;). En este caso habría que indicar el nombre de la clase com.transtools.jdbc.CtsqlJdbcDriver.

2. Establecimiento de la conexión

Una vez registrado el driver ya es posible establecer conexiones con el servidor CTSQL. Para obtener una conexión se debe utilizar el método getConnection de la clase java.sql.DriverManager. Este método a su vez tiene tres signaturas que permiten indicar la información para establecer la conexión de diferentes formas:

public static Connection getConnection(String url, java.util.Properties info) throws SQLException;

Esta signatura permite indicar las variables de entorno, tanto en la URL como dentro de un objeto java.util.Properties, siendo este último el que prevalece.

Ejemplo:

```
String url = "jdbc:ctsql://localhost:20000/almafac";
java.util.Properties props = new java.util.Properties();
props.setProperty( "user", "prueba" );
props.setProperty( "password", "prueba" );
props.setProperty( "DBPATH", "c:\\bases" );
java.sql.Connection con = DriverManager.getConnection( url, props );
```

public static Connection getConnection(String url, String user, String password) throws SQLException;

Esta signatura permite indicar directamente el usuario y la contraseña a utilizar. El resto de variables de entorno se deberán poner dentro de la URL.

Ejemplo:

```
String url = "jdbc:ctsql://localhost:20000/almafac;DBPATH=c:\\bases";
java.sql.Connection con = DriverManager.getConnection( url, "prueba", "prueba" );
```

public static Connection getConnection(String url) throws SQLException;

En esta forma de llamar al método getConnection se debe poner toda la información dentro la URL.

Ejemplo:

```
String url = "jdbc:ctsql://localhost:20000/almafac";
url += ";user=prueba";
url += ";password=prueba";
url += ";DBPATH=c:\\bases";
java.sql.Connection con = DriverManager.getConnection( url );
```

2.1 Observaciones

Para un correcto funcionamiento del driver, y con el fin de evitar la degradación del rendimiento, se recomienda explícitamente cerrar las conexiones y sentencias que se hubiesen creado cuando no se vaya a hacer más uso de ellas. Estos objetos emplean recursos del servidor CTSQL que solo se liberan cuando se destruye el objeto. En Java, esta operación la realiza el *Recolector de Basura*, que no tiene definido cuándo debe liberar el objeto. Por lo tanto, no se puede asegurar que se vayan a liberar los recursos del CTSQL hasta que finalice el programa o se llame explícitamente al método close.

3. Ejemplo

El código Java que se muestra a continuación es un ejemplo de cómo establecer una conexión a un servidor CTSQL haciendo uso del driver JDBC. Una vez obtenido el objeto que implementa la interfaz java.sql.Connection, ya se puede acceder a la base de datos haciendo uso del paquete java.sql.

```
import java.sql.*;
public class CtsqlJdbcTest
{
     private static void registerDriver(String driver)
     {
        try
        {
          Class.forName(driver);
        }
        catch (ClassNotFoundException e)
        {
          System.err.print("ClassNotFoundException: ");
          System.err.println(e.getMessage());
        }
     }

     private static Connection getConnection(String url, String user, String
password)
        throws SQLException
     {
        return DriverManager.getConnection(url, user, password);
     }

     private static String createUrl(String dbHost, String dbService, String
dbPath, String dbName)
     {
        String url = "jdbc:ctsql";
        url += "://" + dbHost;
        if ( (dbService != null) && (dbService.length() > 0) )
          url += ":" + dbService;
        url += "/" + dbName;
        url += ";DBPATH=" + dbPath;
        return url;
     }

     public static void main(String args[])
     {
        String driver = "com.transtools.jdbc.CtsqlJdbcDriver";
        String dbService = "20000";
        String dbHost = "localhost";
        String dbPath = "c:\\cosmos\\projets\\almafac";
        String dbName = "almafac";
        String dbUser = "prueba";
        String dbPassword = "prueba";
        registerDriver( driver );
        String url = createUrl( dbHost, dbService, dbPath, dbName );
```

```
        try
        {
          System.out.println( "Creating connecction to " + url + " ..." );
          Connection connection = getConnection( url, dbUser, dbPassword );
          // A partir de aquí vendría el código JDBC.
          connection.close();
          System.out.println( "Done !!!" );
        }

        catch(SQLException ex)
        {
          System.err.println("SQLException: " + ex.getMessage());
        }
    }
}
```