Álvaro García Fuentes
Horas Libre Configuración
Desarrollo Aplicaciones Web

UD2 A3 COSULTA SELECT

Álvaro García Fuentes
Horas Libre Configuración
Desarrollo Aplicaciones Web

1. Envía un pantallazo de todas las excepciones manejadas por el paquete java.sql

Exception	Description
BatchUpdateException	The subclass of **SQLException** thrown when an error occurs during a batch update operation.
DataTruncation	An exception thrown as a `DataTruncation` exception (on writes) or reported as a `DataTruncation` warning (on reads) when a data values is unexpectedly truncated for reasons other than its having exceeded `MaxFieldSize`.
SQLClientInfoException	The subclass of **SQLException** is thrown when one or more client info properties could not be set on a `Connection`.
SQLDataException	The subclass of **SQLException** thrown when the SQLState class value is '*22*', or under vendor-specified conditions.
SQLException	An exception that provides information on a database access error or other errors.
SQLFeatureNotSupportedException	The subclass of **SQLException** thrown when the SQLState class value is '*0A*' (the value is 'zero' A).
SQLIntegrityConstraintViolationException	The subclass of **SQLException** thrown when the SQLState class value is '*23*', or under vendor-specified conditions.
SQLInvalidAuthorizationSpecException	The subclass of **SQLException** thrown when the SQLState class value is '*28*', or under vendor-specified conditions.
SQLNonTransientConnectionException	The subclass of **SQLException** thrown for the SQLState class value '*08*', or under vendor-specified conditions.
SQLNonTransientException	The subclass of **SQLException** thrown when an instance where a retry of the same operation would fail unless the cause of the `SQLException` is corrected.
SQLRecoverableException	The subclass of **SQLException** thrown in situations where a previously failed operation might be able to succeed if the application performs some recovery steps and retries the entire transaction or in the case of a distributed transaction, the transaction branch.
SQLSyntaxErrorException	The subclass of **SQLException** thrown when the SQLState class value is '*42*', or under vendor-specified conditions.
SQLTimeoutException	The subclass of **SQLException** thrown when the timeout specified by `Statement.setQueryTimeout`, `DriverManager.setLoginTimeout`, `DataSource.setLoginTimeout`,`XADataSource.setLoginTimeout` has expired.
SQLTransactionRollbackException	The subclass of **SQLException** thrown when the SQLState class value is '*40*', or under vendor-specified conditions.
SQLTransientConnectionException	The subclass of **SQLException** for the SQLState class value '*08*', or under vendor-specified conditions.
SQLTransientException	The subclass of **SQLException** is thrown in situations where a previously failed operation might be able to succeed when the operation is retried without any intervention by application-level functionality.
SQLWarning	An exception that provides information on database access warnings.

2. Envía un pantallazo de la relación de herencia entre las excepciones lanzadas por el método getConnection.

```
com.mysql.cj.jdbc.exceptions.CommunicationsException: Communications link failure

The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
        at mysql.connector.java@8.0.26/com.mysql.cj.jdbc.exceptions.SQLError.createCommunicationsException(SQLError.java:174)
        at mysql.connector.java@8.0.26/com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping
        at mysql.connector.java@8.0.26/com.mysql.cj.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:828)
        at mysql.connector.java@8.0.26/com.mysql.cj.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:448)
        at mysql.connector.java@8.0.26/com.mysql.cj.jdbc.ConnectionImpl.getInstance(ConnectionImpl.java:241)
        at mysql.connector.java@8.0.26/com.mysql.cj.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:198)
        at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:677)
        at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:228)
        at ConnectionJDBC.<init>(ConnectionJDBC.java:18)
        at Main.main(Main.java:4)
Caused by: com.mysql.cj.exceptions.CJCommunicationsException: Communications link failure

The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
        at java.base/jdk.internal.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at java.base/jdk.internal.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
        at java.base/jdk.internal.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:4
        at java.base/java.lang.reflect.Constructor.newInstance(Constructor.java:490)
        at mysql.connector.java@8.0.26/com.mysql.cj.exceptions.ExceptionFactory.createException(ExceptionFactory.java:61)
        at mysql.connector.java@8.0.26/com.mysql.cj.exceptions.ExceptionFactory.createException(ExceptionFactory.java:105)
        at mysql.connector.java@8.0.26/com.mysql.cj.exceptions.ExceptionFactory.createException(ExceptionFactory.java:151)
        at mysql.connector.java@8.0.26/com.mysql.cj.exceptions.ExceptionFactory.createCommunicationsException(ExceptionFactory.j
        at mysql.connector.java@8.0.26/com.mysql.cj.protocol.a.NativeSocketConnection.connect(NativeSocketConnection.java:89)
        at mysql.connector.java@8.0.26/com.mysql.cj.NativeSession.connect(NativeSession.java:119)
        at mysql.connector.java@8.0.26/com.mysql.cj.jdbc.ConnectionImpl.connectOneTryOnly(ConnectionImpl.java:948)
        at mysql.connector.java@8.0.26/com.mysql.cj.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:818)
        ... 7 more
Caused by: java.net.ConnectException: Conexión rehusada (Connection refused)
```

Álvaro García Fuentes
Horas Libre Configuración
Desarrollo Aplicaciones Web

```
Caused by: java.net.ConnectException: Conexión rehusada (Connection refused)
        at java.base/java.net.PlainSocketImpl.socketConnect(Native Method)
        at java.base/java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:412)
        at java.base/java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:255)
        at java.base/java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:237)
        at java.base/java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)
        at java.base/java.net.Socket.connect(Socket.java:609)
        at mysql.connector.java@8.0.26/com.mysql.cj.protocol.StandardSocketFactory.connect(StandardSocketFactory.java:156)
        at mysql.connector.java@8.0.26/com.mysql.cj.protocol.a.NativeSocketConnection.connect(NativeSocketConnection.java:63)
        ... 10 more
```

3. La clase DriverManager puede lanzar la SecurityException. Indica cuándo sucede.
Sucede cuando se tiene más de un archivo jar en el classpath.

4. En java el ARM (Automatic Resource Management) es una característica implementada en Java 7. Permite que los recursos que sean Autocloseable se cierren en el orden inverso al que se definen. Averigua qué clases de java.sql son Autocloseables y qué implica que lo sean a la hora de trabajar con ellas.

En java.sql las clases Autoclosables son Connection, CallableStatement, PreparedStatement, ResultSet y Statement. Esto implica que no tendremos que preocuparnos de cerrarlas antes de que nuestro programa termine.

5. ResultSet representa una tabla de datos con los resultados de la base de datos. Normalmente se genera al ejecutar una consulta a la base de datos. Indica:

1. **Cursor. Qué es y a qué apunta**
   Es una variable de un objeto ResultSet que apunta a la fila actual de datos.

2. **Posición inicial del cursor**
   Inicialmente el cursor está posicionado antes de la primera fila de datos.

3. **Getters para recuperar los valores de columna de la fila actual**
   getArray, getAsciiStream, getBigDecimal, getBinaryStream, getBlob, getBoolean, getByte, getBytes, getCharacterStream, getClob, getDate, getDouble, getFloat, getInt, getLong, getNCharacterStream, getNClob, getNString, getObject, getRef, getRowId, getShort, getSQLXML, getString, getTime, getTimeStamp y getURL.

4. **Indica cómo se identifican las columnas**
   Las columnas se identifican bien por un índice (empezando por 1) o por su nombre.

6. Ejecuta una consulta Select original y muestra los datos por salida estándar. Para ello utiliza:

1. Statement statement = con.createStatement());//Se utiliza para implementar sentencias SQL simples sin parámetros.

Álvaro García Fuentes
Horas Libre Configuración
Desarrollo Aplicaciones Web

2. ResultSet rs = statement.executeQuery(query);//Ejecuta una consulta y devuelve un objeto ResultSet. Mediante un cursor/puntero se apunta a una fila de datos
3. while (rs.next()) {//mueve el cursor una fila hacia adelante. Cada vez que llama a next, el método genera los datos en la fila donde está posicionado actualmente el cursor

7. Recuerda enviar los fuentes

En la base de datos MSQL he creado con anterioridad la tabla PROFESORES con el atributo nombre, luego le he añadido varios nombres. Con JDBC hago una consulta para ver el nombre de los profesores mediante este código:

Álvaro García Fuentes
Horas Libre Configuración
Desarrollo Aplicaciones Web

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.SQLTimeoutException;
import java.sql.Statement;

public class ConsultaJDBC {

    public static void main(String[] args){

        String db_ = "HLC";
        String login_ = "alvaro";
        String password_ = "biballo1";
        String url_ = "jdbc:mysql://127.0.0.1:3306/" + db_;
        Connection connection_ = null;
        Statement st_ = null;
        String query = "SELECT nombre FROM PROFESORES;";
        String nombre = null;
        ResultSet rs = null;

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection_ = DriverManager.getConnection(url_, login_, password_);

            if( connection_ != null ) {
                st_ = connection_.createStatement();
                System.out.println( "Conexion a base de datos " + db_ + " correcta." );
                rs = st_.executeQuery(query);
                System.out.println("tabla PROFESORES");
                System.out.println("-nombre-");

                while(rs.next()) {
                    nombre = rs.getString("nombre");
                    System.out.println(nombre + " ");
                }
            } else
                System.out.println("Conexion fallida.");

        } catch ( SQLTimeoutException e )    { System.out.print("Error de tiempo de conexion."); }
          catch ( SQLException e )           { e.printStackTrace(); }
          catch ( ClassNotFoundException e ) { e.printStackTrace(); }
          catch ( Exception e )              { e.printStackTrace (); }
    }
}
```

Y el resultado que nos da el terminal es el siguiente:

```
<terminated> ConsultaJDBC [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java  (29 abr. 2022 21:08:18 – 21:08:19)
Conexion a base de datos HLC correcta.
tabla PROFESORES
-nombre-
eva
juan
justo
laura
```