Java AutoClose

AutoClose

A Java, "AutoClose" és una característica que permet a un objecte tancar-se automàticament una vegada que s'ha completat el seu ús, la qual cosa ajuda a evitar possibles fugides de recursos i a simplificar el codi.

Per a usar AutoClose en una connexió de base de dades, primer has d'assegurar-te que el teu objecte connexió implemente la interfície "AutoCloseable". Després, pots crear un objecte de connexió en un bloc "try-with-resources", la qual cosa assegurarà que es tanque automàticament una vegada que el bloc try haja acabat.

Exemple

Ací hi ha un exemple per a connectar-se a una base de dades MySQL sense AutoClose i amb AutoClose.

En el primer exemple, hem creat els objectes "Connection", "Statement" i "ResultSet" manualment i els hem tancats en el bloc "finally". L'ordre de tancament és important: hem de tancar l'objecte més intern primer (en aquest cas, "ResultSet"), després l'objecte "Statement" i finalment l'objecte "Connection". A més, cada objecte es tanca dins del seu propi bloc try-catch per a manejar qualsevol excepció que puga sorgir.

```
import java.sql.*;

public class DatabaseConnector {
   public static void main(String[] args) {
      String url = "jdbc:mysql://localhost:3306/uf13";
      String user = "root";
      String password = "";
      Connection connection = null;
      Statement statement = null;
      ResultSet resultSet = null;

      try {
         connection = DriverManager.getConnection(url, user, password);
         statement = connection.createStatement();
         resultSet = statement.executeQuery("SELECT * FROM productos");

         while (resultSet.next()) {
            System.out.println(resultSet.getString("column1") + " " +
resultSet.getString("column2"));
         }

      } catch (SQLException e) {
         e.printStackTrace();
      } finally {
         // tanquem manualment els objectes en ordre invers a com es van crear
         if (resultSet != null) {
            try {
               resultSet.close();
            } catch (SQLException e) {
               e.printStackTrace();
            }
         }

         if (statement != null) {
            try {
               statement.close();
            } catch (SQLException e) {
               e.printStackTrace();
            }
         }

         if (connection != null) {
            try {
               connection.close();
            } catch (SQLException e) {
               e.printStackTrace();
            }
         }
      }
   }
}
echo $name;
?>
```

En el segon exemple, podem vore com l'objecte "Connection" es troba dins del bloc try. Per tant, es tancarà automàticament una vegada que el bloc haja acabat, independentment de si s'ha produït o no una excepció.

```
import java.sql.*;

public class DatabaseConnector {
   public static void main(String[] args) {
      String url = "jdbc:mysql://localhost:3306/uf13";
      String user = "root";
      String password = "";

      try (Connection connection = DriverManager.getConnection(url, user, password)) {
         // ací pots fer el que necessites amb la connexió
         Statement statement = connection.createStatement();
         ResultSet resultSet = statement.executeQuery("SELECT * FROM productos");

         while (resultSet.next()) {
            System.out.println(resultSet.getString("column1") + " " + resultSet.getString("column2"));
         }

      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
}
```

Darrera modificació: dilluns, 24 abril 2023, 22:14

« PREVIOUS ACTIVITY
UF13.-Material complementari Exemple 02

NEXT ACTIVITY
Java PreparedStatement »

Salta a...