JDBC - BATCHING WITH STATEMENT OBJECT

Here is a typical sequence of steps to use Batch Processing with Statement Object −

1. Create a Statement object using either *createStatement* methods.
2. Set auto-commit to false using *setAutoCommit*.
3. Add as many as SQL statements you like into batch using *addBatch* method on created statement object.
4. Execute all the SQL statements using *executeBatch* method on created statement object.
5. Finally, commit all the changes using *commit* method.

This sample code has been written based on the environment and database setup done in the previous chapters.

Copy and past the following example in JDBCExample.java, compile and run as follows −

```
// Import required packages
import java.sql.*;

public class JDBCExample {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";

   //  Database credentials
   static final String USER = "username";
   static final String PASS = "password";

   public static void main(String[] args) {
   Connection conn = null;
   Statement stmt = null;
   try{
      // Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");

      // Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL,USER,PASS);

      // Create statement
      System.out.println("Creating statement...");
      stmt = conn.createStatement();

      // Set auto-commit to false
      conn.setAutoCommit(false);

      // First, let us select all the records and display them.
      printRows( stmt );

      // Create SQL statement
      String SQL = "INSERT INTO Employees (id, first, last, age) " +
                   "VALUES(200,'Zia', 'Ali', 30)";
      // Add above SQL statement in the batch.
      stmt.addBatch(SQL);

      // Create one more SQL statement
      SQL = "INSERT INTO Employees (id, first, last, age) " +
            "VALUES(201,'Raj', 'Kumar', 35)";
      // Add above SQL statement in the batch.
      stmt.addBatch(SQL);
      // Create one more SQL statement
      SQL = "UPDATE Employees SET age = 35 " +
            "WHERE id = 100";
      // Add above SQL statement in the batch.
      stmt.addBatch(SQL);

      // Create an int[] to hold returned values
      int[] count = stmt.executeBatch();

      //Explicitly commit statements to apply changes
      conn.commit();

      // Again, let us select all the records and display them.
      printRows( stmt );

      // Clean-up environment
      stmt.close();
      conn.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
      try{
         if(stmt!=null)
            stmt.close();
      }catch(SQLException se2){
      }// nothing we can do
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main

public static void printRows(Statement stmt) throws SQLException{
   System.out.println("Displaying available rows...");
   // Let us select all the records and display them.
   String sql = "SELECT id, first, last, age FROM Employees";
   ResultSet rs = stmt.executeQuery(sql);

   while(rs.next()){
      //Retrieve by column name
      int id  = rs.getInt("id");
      int age = rs.getInt("age");
      String first = rs.getString("first");
      String last = rs.getString("last");

      //Display values
      System.out.print("ID: " + id);
      System.out.print(", Age: " + age);
      System.out.print(", First: " + first);
      System.out.println(", Last: " + last);
   }
   System.out.println();
   rs.close();
}//end printRows()
}//end JDBCExample
```

Now let us compile the above example as follows −

```
C:\>javac JDBCExample.java
C:\>
```

When you run **JDBCExample**, it produces the following result −

```
C:\>java JDBCExample
Connecting to database...
Creating statement...
Displaying available rows...
ID: 95, Age: 20, First: Sima, Last: Chug
ID: 100, Age: 18, First: Zara, Last: Ali
ID: 101, Age: 25, First: Mahnaz, Last: Fatma
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 30, First: Sumit, Last: Mittal
ID: 110, Age: 20, First: Sima, Last: Chug

Displaying available rows...
ID: 95, Age: 20, First: Sima, Last: Chug
ID: 100, Age: 35, First: Zara, Last: Ali
ID: 101, Age: 25, First: Mahnaz, Last: Fatma
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 30, First: Sumit, Last: Mittal
ID: 110, Age: 20, First: Sima, Last: Chug
ID: 200, Age: 30, First: Zia, Last: Ali
ID: 201, Age: 35, First: Raj, Last: Kumar
Goodbye!
C:\>
```