JDBC Database Programming

JDBC - Statement Object Example

Previous Page

Following is the example which makes use of following three queries along with op statment:

boolean execute(String SQL) : Returns a boolean value of true if a Resul retrieved; otherwise, it returns false. Use this method to execute SQL DDL s you need to use truly dynamic SQL.

int executeUpdate(String SQL) : Returns the numbers of rows affected by t SQL statement. Use this method to execute SQL statements for which y number of rows affected - for example, an INSERT, UPDATE, or DELETE state

ResultSet executeQuery(String SQL) : Returns a ResultSet object. Use this expect to get a result set, as you would with a SELECT statement.

This sample code has been written based on the environment and database setu chapters.

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

```
//STEP 1. 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{
      //STEP 2: Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");

      //STEP 3: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL,USER,PASS);

      //STEP 4: Execute a query
      System.out.println("Creating statement...");
      stmt = conn.createStatement();
      String sql = "UPDATE Employees set age=30 WHERE id=103";

      // Let us check if it returns a true Result Set or not.
      Boolean ret = stmt.execute(sql);
```

```
      System.out.println("Return value is : " + ret.toString() );

      // Let us update age of the record with ID = 103;
      int rows = stmt.executeUpdate(sql);
      System.out.println("Rows impacted : " + rows );

      // Let us select all the records and display them.
      sql = "SELECT id, first, last, age FROM Employees";
      ResultSet rs = stmt.executeQuery(sql);

      //STEP 5: Extract data from result set
      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);
      }
      //STEP 6: Clean-up environment
      rs.close();
      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
}//end JDBCExample
```

Now let us compile above example as follows:

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

When you run **JDBCExample**, it produces following result:

```
C:\>java JDBCExample
Connecting to database...
Creating statement...
Return value is : false
Rows impacted : 1
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
Goodbye!
```