

JDBC - SETSAVEPOINT, RELEASESAVEPOINT EXAMPLE

<http://www.tutorialspoint.com/jdbc/jdbc-savepoints.htm>

Copyright © tutorialspoint.com

Following is the example, which makes use of **setSavepoint** and **rollback** described in the Transaction tutorial.

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 –

```
//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: Set auto commit as false.
            conn.setAutoCommit(false);

            //STEP 5: Execute a query to delete statment with
            // required arguments for RS example.
            System.out.println("Creating statement...");
            stmt = conn.createStatement();

            //STEP 6: Now list all the available records.
            String sql = "SELECT id, first, last, age FROM Employees";
            ResultSet rs = stmt.executeQuery(sql);
            System.out.println("List result set for reference....");
            printRs(rs);

            // STEP 7: delete rows having ID grater than 104
            // But save point before doing so.
            Savepoint savepoint1 = conn.setSavepoint("ROWS_DELETED_1");
            System.out.println("Deleting row....");
            String SQL = "DELETE FROM Employees " +
                "WHERE ID = 110";
            stmt.executeUpdate(SQL);
            // oops... we deleted too wrong employees!
            //STEP 8: Rollback the changes afetr save point 2.
            conn.rollback(savepoint1);

            // STEP 9: delete rows having ID grater than 104
            // But save point before doing so.
            Savepoint savepoint2 = conn.setSavepoint("ROWS_DELETED_2");
            System.out.println("Deleting row....");
            SQL = "DELETE FROM Employees " +
                "WHERE ID = 95";
            stmt.executeUpdate(SQL);
```

```

//STEP 10: Now list all the available records.
sql = "SELECT id, first, last, age FROM Employees";
rs = stmt.executeQuery(sql);
System.out.println("List result set for reference....");
printRs(rs);

//STEP 10: Clean-up environment
rs.close();
stmt.close();
conn.close();
}catch(SQLException se){
    //Handle errors for JDBC
    se.printStackTrace();
    // If there is an error then rollback the changes.
    System.out.println("Rolling back data here....");
try{
if(conn!=null)
    conn.rollback();
}catch(SQLException se2){
    se2.printStackTrace();
} //end try

}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 printRs(ResultSet rs) throws SQLException{
    //Ensure we start with first row
    rs.beforeFirst();
    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();
} //end printRs()
} //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...
List result set for reference....
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

Deleting row....
Deleting row....
List result set for reference....
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

Goodbye!
C:\>
```