SQLite Installation

Install SQLite on Windows

- **Step 1** – Go to SQLite download page, and download precompiled binaries from Windows section.
- **Step 2** – Download sqlite-shell-win32-*.zip and sqlite-dll-win32-*.zip zipped files.
- **Step 3** – Create a folder C:\>sqlite and unzip above two zipped files in this folder, which will give you sqlite3.def, sqlite3.dll and sqlite3.exe files.
- **Step 4** – Add C:\>sqlite in your PATH environment variable and finally go to the command prompt and issue sqlite3 command, which should display the following result.

```
C:\>sqlite3
SQLite version 3.7.15.2 2013-01-09 11:53:05
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

JAVA

JDBC Connection

Before you start using SQLite in our Java programs, you need to make sure that you have SQLite JDBC Driver and Java set up on the machine. You can check Java tutorial for Java installation on your machine. Now, let us check how to set up SQLite JDBC driver.

Download latest version of sqlite-jdbc-(VERSION).jar from sqlite-jdbc repository.

Add downloaded jar file sqlite-jdbc-(VERSION).jar in your class path, or you can use it along with -classpath option as explained in the following examples.

Connect to Database

Following Java programs shows how to connect to an existing database. If the database does not exist, then it will be created and finally a database object will be returned.

```
import java.sql.*;

public class SQLiteJDBC {
  public static void main( String args[] ) {
      Connection c = null;

      try {
         Class.forName("org.sqlite.JDBC");
         c = DriverManager.getConnection("jdbc:sqlite:test.db");
      } catch ( Exception e ) {
         System.err.println( e.getClass().getName() + ": " + e.getMessage() );
         System.exit(0);
      }
      System.out.println("Opened database successfully");
   }
}
```

Now, let's compile and run the above program to create our database **test.db** in the current directory. You can change your path as per your requirement. We are assuming the current version of JDBC driver *sqlite-jdbc-3.7.2.jar* is available in the current path.

If you are going to use Windows machine, then you can compile and run your code as follows –

```
$javac SQLiteJDBC.java

$java -classpath ".;sqlite-jdbc-3.7.2.jar" SQLiteJDBC

Opened database successfully
```

Create a Table

Following Java program will be used to create a table in the previously created database.

```
import java.sql.*;
public class SQLiteJDBC {
   public static void main( String args[] ) {
      Connection c = null;
      Statement stmt = null;
      try {
         Class.forName("org.sqlite.JDBC");
         c = DriverManager.getConnection("jdbc:sqlite:test.db");
         System.out.println("Opened database successfully");
         stmt = c.createStatement();
         String sql = "CREATE TABLE COMPANY " +
                        "(ID INT PRIMARY KEY     NOT NULL," +
                        " NAME           TEXT    NOT NULL, " +
                        " AGE            INT     NOT NULL, " +
                        " ADDRESS        CHAR(50), " +
                        " SALARY         REAL)";
         stmt.executeUpdate(sql);
         stmt.close();
         c.close();
      } catch ( Exception e ) {
         System.err.println( e.getClass().getName() + ": " + e.getMessage() );
         System.exit(0);
      }
      System.out.println("Table created successfully");
   }
}
```

When the above program is compiled and executed, it will create COMPANY table in your **test.db** and final listing of the file will be as follows

INSERT Operation

Following Java program shows how to create records in the COMPANY table created in above example.

```
import java.sql.*;
public class SQLiteJDBC {
   public static void main( String args[] ) {
      Connection c = null;
      Statement stmt = null;
      try {
         Class.forName("org.sqlite.JDBC");
         c = DriverManager.getConnection("jdbc:sqlite:test.db");
         c.setAutoCommit(false);
         System.out.println("Opened database successfully");
         stmt = c.createStatement();
         String sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " +
                      "VALUES (1, 'Paul', 32, 'California', 20000.00 );";
         stmt.executeUpdate(sql);
         sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " +
                  "VALUES (2, 'Allen', 25, 'Texas', 15000.00 );";
         stmt.executeUpdate(sql);
         sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " +
                  "VALUES (3, 'Teddy', 23, 'Norway', 20000.00 );";
         stmt.executeUpdate(sql);
         sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) " +
                  "VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 );";
         stmt.executeUpdate(sql);
         stmt.close();
         c.commit();
         c.close();
      } catch ( Exception e ) {
         System.err.println( e.getClass().getName() + ": " + e.getMessage() );
         System.exit(0);
      }
      System.out.println("Records created successfully");
   }
}
```

When above program is compiled and executed, it will create given records in COMPANY table and will display following two line –

```
Opened database successfully
Records created successfully
```

SELECT Operation

Following Java program shows how to fetch and display records from the COMPANY table created in the above example.

```
import java.sql.*;

public class SQLiteJDBC {

  public static void main( String args[] ) {

   Connection c = null;
   Statement stmt = null;
   try {
      Class.forName("org.sqlite.JDBC");
      c = DriverManager.getConnection("jdbc:sqlite:test.db");
      c.setAutoCommit(false);
      System.out.println("Opened database successfully");

      stmt = c.createStatement();
      ResultSet rs = stmt.executeQuery( "SELECT * FROM COMPANY;" );

      while ( rs.next() ) {
         int id = rs.getInt("id");
         String  name = rs.getString("name");
         int age  = rs.getInt("age");
         String  address = rs.getString("address");
         float salary = rs.getFloat("salary");

         System.out.println( "ID = " + id );
         System.out.println( "NAME = " + name );
         System.out.println( "AGE = " + age );
         System.out.println( "ADDRESS = " + address );
         System.out.println( "SALARY = " + salary );
         System.out.println();
      }
      rs.close();
      stmt.close();
      c.close();
   } catch ( Exception e ) {
      System.err.println( e.getClass().getName() + ": " + e.getMessage() );
      System.exit(0);
   }
   System.out.println("Operation done successfully");
  }
}
```

UPDATE Operation

Following Java code shows how to use UPDATE statement to update any record and then fetch and display the updated records from the COMPANY table.

```
import java.sql.*;

public class SQLiteJDBC {

  public static void main( String args[] ) {

   Connection c = null;
   Statement stmt = null;

   try {
      Class.forName("org.sqlite.JDBC");
      c = DriverManager.getConnection("jdbc:sqlite:test.db");
      c.setAutoCommit(false);
      System.out.println("Opened database successfully");

      stmt = c.createStatement();
      String sql = "UPDATE COMPANY set SALARY = 25000.00 where ID=1;";
      stmt.executeUpdate(sql);
      c.commit();

      ResultSet rs = stmt.executeQuery( "SELECT * FROM COMPANY;" );

      while ( rs.next() ) {
         int id = rs.getInt("id");
         String  name = rs.getString("name");
         int age  = rs.getInt("age");
         String  address = rs.getString("address");
         float salary = rs.getFloat("salary");

         System.out.println( "ID = " + id );
         System.out.println( "NAME = " + name );
         System.out.println( "AGE = " + age );
         System.out.println( "ADDRESS = " + address );
         System.out.println( "SALARY = " + salary );
         System.out.println();
      }
      rs.close();
      stmt.close();
      c.close();
   } catch ( Exception e ) {
      System.err.println( e.getClass().getName() + ": " + e.getMessage() );
      System.exit(0);
   }
    System.out.println("Operation done successfully");
   }
}
```

DELETE Operation

Following Java code shows how to use use DELETE statement to delete any record and then fetch and display the remaining records from the our COMPANY table.

```
import java.sql.*;

public class SQLiteJDBC {

   public static void main( String args[] ) {
      Connection c = null;
      Statement stmt = null;

      try {
         Class.forName("org.sqlite.JDBC");
         c = DriverManager.getConnection("jdbc:sqlite:test.db");
         c.setAutoCommit(false);
         System.out.println("Opened database successfully");

         stmt = c.createStatement();
         String sql = "DELETE from COMPANY where ID=2;";
         stmt.executeUpdate(sql);
         c.commit();

         ResultSet rs = stmt.executeQuery( "SELECT * FROM COMPANY;" );

         while ( rs.next() ) {
         int id = rs.getInt("id");
         String  name = rs.getString("name");
         int age  = rs.getInt("age");
         String  address = rs.getString("address");
         float salary = rs.getFloat("salary");

         System.out.println( "ID = " + id );
         System.out.println( "NAME = " + name );
         System.out.println( "AGE = " + age );
         System.out.println( "ADDRESS = " + address );
         System.out.println( "SALARY = " + salary );
         System.out.println();
      }
      rs.close();
      stmt.close();
      c.close();
      } catch ( Exception e ) {
         System.err.println( e.getClass().getName() + ": " + e.getMessage() );
         System.exit(0);
      }
      System.out.println("Operation done successfully");
   }
}
```