tutlane (/)

# SQLite JAVA Tutorial

Here we will learn how to use SQLite in JAVA programming language to connect SQLite database, CREATE (/tutorial/sqlite/sqlite-create-table) Table, INSERT (/tutorial/sqlite/sqlite-insert-query), UPDATE (/tutorial/sqlite/sqlite-update-statement), DELETE (/tutorial/sqlite/sqlite-delete-statement) and SELECT (/tutorial/sqlite/sqlite-select-query) operations on SQLite tables using JDBC driver with examples.

## SQLite in JAVA Interface

We can easily interact with SQLite in JAVA language using a JDBC driver. This JDBC driver is known as the SQLite-JDBC package which contains both JAVA classes and SQLite libraries to perform different operations like connect to the database, create tables, insert data in tables, etc. on Windows, Linux, and Mac OS platform.

Before we proceed to interact with SQLite using JAVA language first we need to make sure that JAVA setup available in our PC. In case if JAVA setup is not available means follow Java Tutorial for JAVA installation.

If JAVA setup available in our PC, now we will install SQLite-JDBC driver for that download latest JDBC driver sqlite-jdbc-version.jar from available list of JDBC (https://bitbucket.org/xerial/sqlite-jdbc/downloads/) drivers.

Now we need to add downloaded JDBC driver jar file (**sqlite-jdbc-version.jar**) to our classpath like shown in our following programs.

## Connect to SQLite Database using Java

Now we will connect to the SQLite database using JAVA if exists otherwise it will create a new database and then connect to it.

Following is the JAVA program which is used to connect a database if it exists otherwise first it will create a database and then connect to it.

```
import java.sql.*;
public class DBUsingJava
{
public static void main( String args[] )
{
Connection c = null;
try {
Class.forName("org.sqlite.JDBC");
c = DriverManager.getConnection("jdbc:sqlite:SqliteJavaDB.db");
}
catch ( Exception e ) {
```

```
System.err.println( e.getClass().getName() + ": " + e.getMessage() );
System.exit(0);
}
System.out.println("database successfully created");
}
}
```

If you observe above code we are trying to connect "**SqliteJavaDB.db**" if exists otherwise it will create new database in current path and we are assuming that **sqlite-jdbc-3.8.11.2.jar** is available at the same location where our program exists.

```
Javac DBUsingJava.java
java -classpath ".;sqlite-jdbc-3.8.11.2.jar" SQLiteJDBC

database successfully created
```

The above statements will compile and run our program to create "**SqliteJavaDB.db**" in current directory. If you check the current directory of the program, one file called **SqliteJavaDB.db** created.

## Create Table in SQLite Database using Java

Now, we will create new table in previously created database named **SqliteJavaDB.db** using java for that write the code like as shown below.

```
import java.sql.*;
public class TableUsingJava
{
public static void main( String args[] )
{
Connection c = null;
Statement stmt = null;
try {
Class.forName("org.sqlite.JDBC");
c = DriverManager.getConnection("jdbc:sqlite:SqliteJavaDB.db");
System.out.println("Database Opened...\n");
stmt = c.createStatement();
String sql = "CREATE TABLE Product " +
"(p_id INTEGER PRIMARY KEY AUTOINCREMENT," +
" p_name TEXT NOT NULL, " +
" price REAL NOT NULL, " +
" quantity INTEGER) " ;
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 Product Created Successfully!!!");
}
}
```

If you observe above code we are creating a new table called “**Product**” in our “**SqliteJavaDB.db**” database.

Now compile and run the program using the following commands.

```
javac TableUsingJava.java
java -classpath ".;sqlite-jdbc-3.8.11.2.jar" TableUsingJava
Database Opened...

Table Product Created Successfully!!!
```

The above statements complied with our program and created “Product” table in our database.

## DML (Insert, Select, Update, Delete) Operations Using Java

Now, we try to perform DML operations (insert, select, update, delete) on previously created table named Product using JAVA language.

Following program contains all 4 DML operations like INSERT (/tutorial/sqlite/sqlite-insert-query), UPDATE (/tutorial/sqlite/sqlite-update-statement), DELETE (/tutorial/sqlite/sqlite-delete-statement) and SELECT (/tutorial/sqlite/sqlite-select-query).

```
import java.util.Scanner;
import java.sql.*;
public class OperationUsingJava
{
public static void main( String args[] )
{
String flag="Y";
do{
System.out.println("Select DML Operation For Product Table...");
System.out.println("1. Insert");
System.out.println("2. Update");
System.out.println("3. Delete");
System.out.println("4. Select");
System.out.println("5. Exit");
Scanner reader = new Scanner(System.in);
System.out.println("Enter a choice: ");
```

```
int n = reader.nextInt();
Connection c = null;
Statement stmt = null;
try {
Class.forName("org.sqlite.JDBC");
c = DriverManager.getConnection("jdbc:sqlite:SqliteJavaDB.db");
c.setAutoCommit(false);
stmt = c.createStatement();
String name="",sql="";
float price=0.0f;
int quantity=0;
int id;
Scanner scanName;
switch(n){

case 1:
scanName=new Scanner(System.in);
System.out.println("Enter Product Name:");
name=scanName.nextLine();
System.out.println("Enter Product Price:");
price=scanName.nextFloat();
System.out.println("Enter Product Quantity:");
quantity=scanName.nextInt();
sql = "INSERT INTO Product (p_name,price,quantity) " +
"VALUES ('" +name+ "'," +
price + "," + quantity + ")";
stmt.executeUpdate(sql);
System.out.println("Inserted Successfully!!!");
break;

case 2:
System.out.println("Enter Product id:");
scanName=new Scanner(System.in);
id=scanName.nextInt();
System.out.println("Enter Product Name:");
scanName=new Scanner(System.in);
name=scanName.nextLine();
System.out.println("Enter Product Price:");
price=scanName.nextFloat();
System.out.println("Enter Product Quantity:");
quantity=scanName.nextInt();

sql = "UPDATE Product SET p_name = '"+ name + "',price=" + price +",quantity=" + q
uantity +
" WHERE p_id=" +id ;
stmt.executeUpdate(sql);
System.out.println("Updated Successfully!!!");
break;

case 3:
System.out.println("Enter Product id:");
scanName=new Scanner(System.in);
id=scanName.nextInt();
sql="DELETE FROM Product WHERE p_id=" + id+";";
stmt.executeUpdate(sql);
System.out.println("Deleted Successfully!!!");
break;

case 4:
ResultSet rs = stmt.executeQuery("SELECT * FROM Product;");
System.out.println("ID\t Name\t\t Price\t Qty ");
while ( rs.next() ) {
id = rs.getInt("p_id");
name = rs.getString("p_name");
quantity = rs.getInt("quantity");
price = rs.getFloat("price");
System.out.println(id+"\t "+name+" \t "+price+"\t "+quantity);
}
rs.close();
break;

case 5:
System.exit(0);
break;

default:
System.out.println("Oops!!! Wrong Choice...");
break;
}
stmt.close();
c.commit();
c.close();
}
catch ( Exception e )
{
System.err.println( e.getClass().getName() + ": " + e.getMessage() );
System.exit(0);
}
```

```
System.out.println("Continue Y OR N?");
reader=new Scanner(System.in);
flag=reader.nextLine();

}while(flag.equalsIgnoreCase("Y"));
System.exit(0);
}
}
```

If you observe above program we are performing INSERT (/tutorial/sqlite/sqlite-insert-query), UPDATE (/tutorial/sqlite/sqlite-update-statement), DELETE (/tutorial/sqlite/sqlite-delete-statement) and SELECT (/tutorial/sqlite/sqlite-select-query) operations on table called "**Product**". Now, let's compile and run the program to examine the output like as shown below.

```
javac OperationUsingJava.java
java -classpath ".;sqlite-jdbc-3.8.11.2.jar" OperationUsingJava

Select DML Operation For Product Table...
1. Insert
2. Update
3. Delete
4. Select
5. Exit

Enter a choice:
1

Enter Product Name:
Pencil
Enter Product Price:
5
Enter Product Quantity:
50

Inserted Successfully!!!

Continue Y OR N?
Y

Select DML Operation For Product Table...
1. Insert
2. Update
3. Delete
4. Select
5. Exit
```

```
Enter a choice:
4

ID Name Price Qty
---- ------ ----- ----
1 Pencil 5.0 50

Continue Y OR N?
Y

Select DML Operation For Product Table...
1. Insert
2. Update
3. Delete
4. Select
5. Exit

Enter a choice:
2

Enter Product id:
1
Enter Product Name:
Sharpner
Enter Product Price:
10
Enter Product Quantity:
90

Updated Successfully!!!

Continue Y OR N?
Y

Select DML Operation For Product Table...
1. Insert
2. Update
3. Delete
4. Select
5. Exit

Enter a choice:
4
```

```
ID Name Price Qty
---- -------- ----- ----
1 Sharpner 10.0 90

Continue Y OR N?
Y

Select DML Operation For Product Table...
1. Insert
2. Update
3. Delete
4. Select
5. Exit

Enter a choice:
1

Enter Product Name:
Scale
Enter Product Price:
5
Enter Product Quantity:
60

Inserted Successfully!!!

Continue Y OR N?
Y

Select DML Operation For Product Table...
1. Insert
2. Update
3. Delete
4. Select
5. Exit

Enter a choice:
4

ID Name Price Qty
---- -------- ---- ---
1 Sharpner 10.0 90
2 Scale 5.0 60

Continue Y OR N?
```

```
Y

Select DML Operation For Product Table...
1. Insert
2. Update
3. Delete
4. Select
5. Exit

Enter a choice:
3

Enter Product id:
2

Deleted Successfully!!!

Continue Y OR N?
y

Select DML Operation For Product Table...
1. Insert
2. Update
3. Delete
4. Select
5. Exit

Enter a choice:
4

ID Name Price Qty
---- --------- ----- ----
1 Sharpner 10.0 90

Continue Y OR N?
n
```

This is how we can use SQLite database in JAVA to perform INSERT (/tutorial/sqlite/sqlite-insert-query), UPDATE (/tutorial/sqlite/sqlite-update-statement), DELETE (/tutorial/sqlite/sqlite-delete-statement) and SELECT (/tutorial/sqlite/sqlite-select-query) operations based on our requirements.