Apache Commons DBUtils

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com


https://www.facebook.com/tutorialspointindia


https://twitter.com/tutorialspoint

About the Tutorial

Apache Commons DBUtils library is a quite small set of classes, which are designed to make easier JDBC call processing without resource leak and to have cleaner code. This tutorial covers most of the topics required for a basic understanding of Apache Commons DBUtils and to get a feel of how it works.

Audience

This tutorial has been prepared for the beginners to help them understand the basic to advanced concepts related to Apache Commons DBUtils.

Prerequisites

Before you start practicing various types of examples given in this reference, we assume that you are already aware about the computer programs and computer programming languages.

Copyright & Disclaimer

Table of Contents

1. Apache Commons DBUtils — Overview

Apache Commons DbUtils library is a quite small set of classes, which are designed to make easier JDBC call processing without resource leak and to have cleaner code. As JDBC resource cleanup is quite tedious and error prone, DBUtils classes helps to abstract out the boiler plate code, so that the developers can focus on database related operations only.

Advantages of DBUtils

The advantages of using Apache Commons DBUtils are explained below:

- **No Resource Leakage** – DBUtils classes ensures that no resource leakage happen.
- **Clean & Clear code** – DBUtils classes provides clean and clear code to do the database operations without any need to write a cleanup or resource leak prevention code.
- **Bean Mapping** – DBUtils class supports to automatically populate javabeans from a result set.

DBUtils Design Principles

The design principles of Apache Commons DBUtils are as follows:

- **Small** – DBUtils library is very small in size with fewer classes, so that it is easy to understand and use.
- **Transparent** – DBUtils library is not doing much work behind the scenes. It simply takes query and executes.
- **Fast** – DBUtils library classes do not create many background objects and is quite fast in database operation executions.

2. Apache Commons DBUtils — Environment Setup

To start developing with DBUtils, you should setup your DBUtils environment by following the steps given below. We assume that you are working on a Windows platform.

Install Java

Install J2SE Development Kit 5.0 (JDK 5.0) from Java Official Site, which is available at https://www.oracle.com/java/technologies/.

Make sure that the following environment variables are set as described below:

- **JAVA_HOME** – This environment variable should point to the directory, where you have installed the JDK. For example, **C:\Program Files\Java\jdk1.5.0**.
- **CLASSPATH** – This environment variable should have appropriate paths set. For example, **C:\Program Files\Java\jdk1.5.0_20\jre\lib**.
- **PATH** – This environment variable should point to appropriate JRE bin. For example, **C:\Program Files\Java\jre1.5.0_20\bin**.

It is possible that you have these variable set already, but just to make sure here's given below how to check, if you have the variable set or not.

- Go to the control panel and double-click on System. If you are a Windows XP user, it is possible that you have to open Performance and Maintenance, before you will see the System icon.
- Go to the Advanced tab and click on the Environment Variables.
- Now, check if all the above mentioned variables are set properly.

Install Database

The most important thing you will need, of course, is an actual running database with a table that you can query and modify.

Install a database that is most suitable for you. You can have plenty of choices and most common choices are stated below:

MySQL DB

MySQL is an open source database. You can download it from **MySQL Official Site**, available at https://dev.mysql.com/downloads/mysql/. We recommend downloading the full Windows installation.

In addition, download and install **MySQL Administrator** from https://dev.mysql.com/downloads/workbench/ as well as **MySQL Query Browser** from https://dev.mysql.com/downloads/workbench/ These are GUI based tools that will make your development much easier.

Finally, download and unzip **MySQL Connector/J** (the MySQL JDBC driver) from https://dev.mysql.com/downloads/connector/j/3.1.html in a convenient directory. For the purpose of this tutorial, we will assume that you have installed the driver at **C:\Program Files\MySQL\mysql-connector-java-5.1.8.**

Accordingly, set CLASSPATH variable to **C:\Program Files\MySQL\mysql-connector-java-5.1.8\mysql-connector-java-5.1.8-bin.jar.** Your driver version may vary based on your installation.

PostgreSQL DB

PostgreSQL is an open source database. You can download it from **PostgreSQL Official Site,** which is available at https://www.postgresql.org/download/.

The Postgres installation contains a GUI based administrative tool called pgAdmin III. JDBC drivers are also included as part of the installation.

Oracle DB

Oracle DB is a commercial database sold by Oracle. We assume that you have the necessary distribution media to install it.

Oracle installation includes a GUI based administrative tool called Enterprise Manager. JDBC drivers are also included as a part of the installation.

Install Database Drivers

The latest JDK includes a JDBC-ODBC Bridge driver that makes most Open Database Connectivity (ODBC) drivers available to programmers using the JDBC API.

Now-a-days, most of the Database vendors are supplying appropriate JDBC drivers along with Database installation. So, you should not worry about this part.

Set Database Credential

For this tutorial, we are going to use MySQL database. When you install any of the above database, its administrator ID is set to **root** and gives provision to set a password of your choice.

By using root ID and password, you can either create another user ID and password, or you can use root ID and password for your JDBC application.

There are various database operations like database creation and deletion, which would need administrator ID and password.

For rest of the JDBC tutorial, we would use MySQL Database with **username** as ID and **password** as password.

If you do not have sufficient privilege to create new users, then you can ask your Database Administrator (DBA) to create a user ID and password for you.

Create Database

To create the **emp** database, use the following steps:

Step 1

Open a **Command Prompt** and change to the installation directory as follows:

```
C:\>
C:\>cd Program Files\MySQL\bin
C:\Program Files\MySQL\bin>
```

Note: The path to **mysqld.exe** may vary depending on the install location of MySQL on your system. You can also check documentation on how to start and stop your database server.

Step 2

Start the database server by executing the following command, if it is already not running.

```
C:\Program Files\MySQL\bin>mysqld
C:\Program Files\MySQL\bin>
```

Step 3

Create the **emp** database by executing the following command:

```
C:\Program Files\MySQL\bin> mysqladmin create emp -u root -p
Enter password: ********
C:\Program Files\MySQL\bin>
```

Create Table

To create the **Employees** table in emp database, use the following steps:

Step 1

Open a **Command Prompt** and change to the installation directory as follows:

```
C:\>
C:\>cd Program Files\MySQL\bin
C:\Program Files\MySQL\bin>
```

Step 2

Login to the database as follows:

```
C:\Program Files\MySQL\bin>mysql -u root -p
Enter password: ********
mysql>
```

Step 3

Create the table **Employee** as follows:

```
mysql> use emp;
mysql> create table Employees
    -> (
    -> id int not null,
    -> age int not null,
    -> first varchar (255),
    -> last varchar (255)
    -> );
Query OK, 0 rows affected (0.08 sec)
mysql>
```

Create Data Records

Finally, you create few records in Employee table as follows:

```
mysql> INSERT INTO Employees VALUES (100, 18, 'Zara', 'Ali');
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO Employees VALUES (101, 25, 'Mahnaz', 'Fatma');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (102, 30, 'Zaid', 'Khan');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (103, 28, 'Sumit', 'Mittal');
Query OK, 1 row affected (0.00 sec)

mysql>
```

For a complete understanding on MySQL database, study the **MySQL Tutorial**, available at https://www.tutorialspoint.com/mysql/index.htm.

Download Commons DBUtils Archive

- Download the latest version of Apache Common DBUtils jar file from **commons-dbutils-1.7-bin.zip** available at https://commons.apache.org/proper/commons-dbutils/download_dbutils.cgi.

- MySql connector **mysql-connector-java-5.1.28-bin.jar** available at https://mvnrepository.com/artifact/mysql/mysql-connector-java.
- Apache Commons DBCP **commons-dbcp2-2.1.1-bin.zip** from https://commons.apache.org/proper/commons-dbcp/download_dbcp.cgi.
- Apache Commons Pool **commons-pool2-2.4.3-bin.zip** from https://commons.apache.org/proper/commons-pool/download_pool.cgi.
- Apache Commons Logging **commons-logging-1.2-bin.zip** available at https://commons.apache.org/proper/commons-logging/download_logging.cgi.

At the time of writing this tutorial, we have downloaded **commons-dbutils-1.7-bin.zip, mysql-connector-java-5.1.28-bin.jar**, **commons-dbcp2-2.1.1-bin.zip**, **commons-pool2-2.4.3-bin.zip**, **commons-logging-1.2-bin.zip** and copied it into **C:\>Apache folder.**

OS	Archive name
Windows	commons-dbutils-1.7-bin.zip
Linux	commons-dbutils-1.7-bin.tar.gz
Mac	commons-dbutils-1.7-bin.tar.gz

Apache Common DBUtils Environment

Set the **APACHE_HOME** environment variable to point to the base directory location, where Apache jar is stored on your machine. Assume that we have extracted **commons-dbutils-1.7-bin.zip** in Apache folder on various Operating Systems as follows:

OS	Output
Windows	Set the environment variable APACHE_HOME to C:\Apache
Linux	export APACHE_HOME=/usr/local/Apache
Mac	export APACHE_HOME=/Library/Apache

Set CLASSPATH Variable

Set the **CLASSPATH** environment variable to point to the Common IO jar location. Assume that you have stored **commons-dbutils-1.7-bin.zip** in Apache folder on various Operating Systems as follows:

OS	Output
Windows	Set the environment variable CLASSPATH to %CLASSPATH%;%APACHE_HOME%\commons-dbutils-1.7.jar;mysql-connector-java-5.1.28.jar;commons-dbcp2-2.1.1.jar;commons-pool2-2.4.3.jar;commons-logging-1.2.jar;
Linux	export CLASSPATH=$CLASSPATH:$APACHE_HOME/commons-dbutils-1.7.jar:mysql-connector-java-5.1.28.jar:commons-dbcp2-2.1.1:commons-pool2-2.4.3.jar:commons-logging-1.2.jar.
Mac	export CLASSPATH=$CLASSPATH:$APACHE_HOME/commons-dbutils-1.7.jar:mysql-connector-java-5.1.28:commons-dbcp2-2.1.1.jar:commons-pool2-2.4.3.jar;commons-logging-1.2.jar.

Now, you are ready to start experimenting with DBUtils. The next chapter gives you a sample example on DBUtils Programming.

3. Apache Commons DBUtils — First Application

This chapter provides an example of how to create a simple JDBC application using DBUtils library. This will show you, how to open a database connection, execute a SQL query, and display the results.

All the steps mentioned in this template example, would be explained in subsequent chapters of this tutorial.

Creating JDBC Application

There are following six steps involved in building a JDBC application:

- **Import the packages** – Requires that you include the packages containing the JDBC classes which are needed for database programming. Most often, using **import java.sql.*** will suffice.
- **Register the JDBC driver** – Requires that you initialize a driver, so you can open a communication channel with the database.
- **Open a connection** – Requires using the **DriverManager.getConnection()** method to create a Connection object, which represents a physical connection with the database.
- **Execute a query** – Requires using an object of type Statement for building and submitting an SQL statement to the database.
- **Extract data from result set** – Requires that you use the appropriate **ResultSet.getXXX()** method to retrieve the data from the result set.
- **Clean up the environment** – Requires explicitly closing all the database resources versus relying on the JVM's garbage collection.

Sample Code

This sample example can serve as a **template,** when you need to create your own JDBC application in the future.

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

Copy and paste the following example in MainApp.java, compile and run as follows:

MainApp.java

```
import java.sql.Connection;
import java.sql.DriverManager;
```

```
import java.sql.SQLException;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.ResultSetHandler;
import org.apache.commons.dbutils.handlers.BeanHandler;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";

   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();

      //Step 1: Register JDBC driver
      DbUtils.loadDriver(JDBC_DRIVER);

      //Step 2: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);

      //Step 3: Create a ResultSet Handler to handle Employee Beans
      ResultSetHandler<Employee> resultHandler = new
BeanHandler<Employee>(Employee.class);

      try {
         Employee emp = queryRunner.query(conn, "SELECT * FROM employees WHERE
first=?",
            resultHandler, "Sumit");
         //Display values
         System.out.print("ID: " + emp.getId());
```

```
            System.out.print(", Age: " + emp.getAge());
            System.out.print(", First: " + emp.getFirst());
            System.out.println(", Last: " + emp.getLast());
         } finally {
            DbUtils.close(conn);
         }
      }
   }
```

Employee.java

The program is given below:

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
```

```
   }

   public void setLast(String last) {
      this.last = last;
   }
}
```

Now let us compile the above example as follows:

```
C:\>javac MainApp.java Employee.java
C:\>
```

When you run **MainApp**, it produces the following result:

```
C:\>java MainApp
Connecting to database...
ID: 103, Age: 28, First: Sumit, Last: Mittal
C:\>
```

4. Apache Commons DBUtils — Create Query

The following example will demonstrate how to create a record using Insert query with the help of DBUtils. We will insert a record in Employees Table.

Syntax

The syntax to create a query is given below:

```
String insertQuery ="INSERT INTO employees(id,age,first,last)  VALUES
(?,?,?,?)";

int insertedRecords = queryRunner.update(conn, insertQuery,104,30,
"Sohan","Kumar");
```

Where,

- **insertQuery** – Insert query having placeholders.
- **queryRunner** – QueryRunner object to insert employee object in database.

To understand the above-mentioned concepts related to DBUtils, let us write an example which will run an insert query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.
2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
   public void setId(int id) {
```

```
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
```

```
   static final String PASS = "admin";

   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();
      DbUtils.loadDriver(JDBC_DRIVER);
      conn = DriverManager.getConnection(DB_URL, USER, PASS);
      try {
         int insertedRecords = queryRunner.update(conn,
            "INSERT INTO employees(id,age,first,last)  VALUES (?,?,?,?)",
            104,30, "Sohan","Kumar");
         System.out.println(insertedRecords + " record(s) inserted");
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
1 record(s) inserted.
```

5. Apache Commons DBUtils — Read Query

The following example will demonstrate how to read a record using Read query with the help of DBUtils. We will read a record from Employees Table.

Syntax

The syntax for read query is mentioned below:

```
ResultSetHandler<Employee> resultHandler = new
BeanHandler<Employee>(Employee.class);

Employee emp = queryRunner.query(conn, "SELECT * FROM employees WHERE first=?",
resultHandler, "Sumit");
```

Where,

- **resultHandler** – ResultSetHandler object to map the result set to Employee object.
- **queryRunner** – QueryRunner object to read an employee object from database.

To understand the above mentioned concepts related to DBUtils, let us write an example which will run a read query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.
2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
```

```
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.ResultSetHandler;
import org.apache.commons.dbutils.handlers.BeanHandler;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";
```

```
   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";

   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();
      //Step 1: Register JDBC driver
      DbUtils.loadDriver(JDBC_DRIVER);

      //Step 2: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);

      //Step 3: Create a ResultSet Handler to handle Employee Beans
      ResultSetHandler<Employee> resultHandler =
         new BeanHandler<Employee>(Employee.class);

      try {
         Employee emp = queryRunner.query(conn, "SELECT * FROM employees WHERE
id=?",
            resultHandler, 104);
         //Display values
         System.out.print("ID: " + emp.getId());
         System.out.print(", Age: " + emp.getAge());
         System.out.print(", First: " + emp.getFirst());
         System.out.println(", Last: " + emp.getLast());
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
ID: 104, Age: 30, First: Sohan, Last: Kumar
```

6. Apache Commons DBUtils — Update Query

The following example will demonstrate how to update a record using Update query with the help of DBUtils. We'll update a record in Employees Table.

Syntax

The syntax for update query is as follows:

```
String updateQuery = "UPDATE employees SET age=? WHERE id=?";
int updatedRecords = queryRunner.update(conn, updateQuery, 33,104);
```

Where,

- **updateQuery** – Update query having placeholders.
- **queryRunner** – QueryRunner object to update employee object in database.

To understand the above mentioned concepts related to DBUtils, let us write an example which will run an update query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.
2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
```

```
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";
```

```
   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();

      DbUtils.loadDriver(JDBC_DRIVER);
      conn = DriverManager.getConnection(DB_URL, USER, PASS);
      try {
         int updatedRecords = queryRunner.update(conn,
            "UPDATE employees SET age=? WHERE id=?", 33,104);
         System.out.println(updatedRecords + " record(s) updated.");
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
1 record(s) updated.
```

7. Apache Commons DBUtils — Delete Query

The following example will demonstrate how to delete a record using Delete query with the help of DBUtils. We will delete a record in Employees Table.

Syntax

The syntax for delete query is mentioned below:

```
String deleteQuery = "DELETE FROM employees WHERE id=?";
int deletedRecords = queryRunner.delete(conn, deleteQuery, 33,104);
```

Where,

- **deleteQuery** – DELETE query having placeholders.
- **queryRunner** – QueryRunner object to delete employee object in database.

To understand the above-mentioned concepts related to DBUtils, let us write an example which will run a delete query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.
2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
```

```
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";
```

```
   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();

      DbUtils.loadDriver(JDBC_DRIVER);
      conn = DriverManager.getConnection(DB_URL, USER, PASS);
      try {
         int deletedRecords = queryRunner.update(conn,
            "DELETE from employees WHERE id=?", 104);
         System.out.println(deletedRecords + " record(s) deleted.");
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
1 record(s) deleted.
```

8. Apache Commons DBUtils - QueryRunner interface

The **org.apache.commons.dbutils.QueryRunner** class is the central class in the DBUtils library. It executes SQL queries with pluggable strategies for handling ResultSets. This class is thread safe.

Class Declaration

Following is the declaration for org.apache.commons.dbutils.QueryRunner class:

```
public class QueryRunner
   extends AbstractQueryRunner
```

Usage

- **Step 1** – Create a connection object.
- **Step 2** – Use QueryRunner object methods to make database operations.

Example

Following example will demonstrate how to read a record using QueryRunner class. We will read one of the available record in employee Table.

Syntax

The syntax is given below:

```
ResultSetHandler<Employee> resultHandler = new
BeanHandler<Employee>(Employee.class);

Employee emp =

   queryRunner.query(conn, "SELECT * FROM employees WHERE first=?",
resultHandler, "Sumit");
```

Where,

- **resultHandler** – ResultSetHandler object to map result set to Employee object.
- **queryRunner** – QueryRunner object to read employee object from database.

To understand the above-mentioned concepts related to DBUtils, let us write an example which will run a read query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.

2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.ResultSetHandler;
import org.apache.commons.dbutils.handlers.BeanHandler;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";

   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();

      //Step 1: Register JDBC driver
      DbUtils.loadDriver(JDBC_DRIVER);

      //Step 2: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);

      //Step 3: Create a ResultSet Handler to handle Employee Beans
      ResultSetHandler<Employee> resultHandler = new
BeanHandler<Employee>(Employee.class);

      try {
         Employee emp = queryRunner.query(conn, "SELECT * FROM employees WHERE
id=?",
```

```
            resultHandler, 103);
         //Display values
         System.out.print("ID: " + emp.getId());
         System.out.print(", Age: " + emp.getAge());
         System.out.print(", First: " + emp.getFirst());
         System.out.println(", Last: " + emp.getLast());
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
ID: 103, Age: 28, First: Sumit, Last: Mittal
```

9. Apache Commons DBUtils — AsyncQueryRunner interface

The **org.apache.commons.dbutils.AsyncQueryRunner** class helps to execute long running SQL queries with async support. This class is thread safe. This class supports same methods as QueryRunner, but it returns Callable objects which can be used later to retrieve the result.

Class Declaration

Following is the declaration for org.apache.commons.dbutils.AsyncQueryRunner class:

```
public class AsyncQueryRunner
   extends AbstractQueryRunner
```

Usage

- **Step 1** – Create a connection object.
- **Step 2** – Use AsyncQueryRunner object methods to make database operations.

Example

Following example will demonstrate how to update a record using AsyncQueryRunner class. We will update one of the available record in employee Table.

Syntax

The syntax is mentioned below:

```
String updateQuery = "UPDATE employees SET age=? WHERE id=?";
future = asyncQueryRunner.update(conn,
            "UPDATE employees SET age=? WHERE id=?", 33,103);
```

Where,

- **updateQuery** – Update query having placeholders.
- **asyncQueryRunner** – asyncQueryRunner object to update employee object in database.
- **future** – Future object to retrieve result later.

To understand the above-mentioned concepts related to DBUtils, let us write an example which will run an update query in async mode. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.
2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
```

```
      this.last = last;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.commons.dbutils.AsyncQueryRunner;
import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;

import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorCompletionService;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";

   public static void main(String[] args) throws
      SQLException, InterruptedException,
      ExecutionException, TimeoutException {
      Connection conn = null;

      AsyncQueryRunner asyncQueryRunner = new AsyncQueryRunner(
```

```
Executors.newCachedThreadPool());

      DbUtils.loadDriver(JDBC_DRIVER);
      conn = DriverManager.getConnection(DB_URL, USER, PASS);
      Future<Integer> future = null;
      try {

         future = asyncQueryRunner.update(conn,
            "UPDATE employees SET age=? WHERE id=?", 33,103);

         Integer updatedRecords = future.get(10, TimeUnit.SECONDS);
         System.out.println(updatedRecords + " record(s) updated.");
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message.

```
1 record(s) updated.
```

10. Apache Commons DBUtils — ResultSetHandler interface

The **org.apache.commons.dbutils.ResultSetHandler** interface is responsible to convert ResultSets into objects.

Class Declaration

Following is the declaration for org.apache.commons.dbutils.ResultSetHandler class

```
public interface ResultSetHandler<T>
```

Usage

- **Step 1** – Create a connection object.
- **Step 2** – Create implementation of ResultSetHandler.
- **Step 3** – Pass resultSetHandler to QueryRunner object, and make database operations.

Example

Following example will demonstrate how to map a record using ResultSetHandler class. We will read one of the available record in Employee Table.

Syntax

The syntax is mentioned below:

```
Employee emp = queryRunner.query(conn, "SELECT * FROM employees WHERE first=?",
resultHandler, "Sumit");
```

Where,

- **resultHandler** – ResultSetHandler object to map result set to Employee object.
- **queryRunner** – QueryRunner object to read employee object from database.

To understand the above-mentioned concepts related to DBUtils, let us write an example which will run a read query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.
2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

```
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.util.Arrays;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.ResultSetHandler;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";

   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();

      //Step 1: Register JDBC driver
      DbUtils.loadDriver(JDBC_DRIVER);

       //Step 2: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);

      //Step 3: Create a ResultSet Handler to handle Employee Beans
      ResultSetHandler<Object[]> handler = new ResultSetHandler<Object[]>() {
         public Object[] handle(ResultSet rs) throws SQLException {
            if (!rs.next()) {
               return null;
            }
            ResultSetMetaData meta = rs.getMetaData();
            int cols = meta.getColumnCount();
            Object[] result = new Object[cols];
```

```
            for (int i = 0; i < cols; i++) {
               result[i] = rs.getObject(i + 1);
            }
            return result;
         }
      };

      try {
         Object[] result  = queryRunner.query(conn, "SELECT * FROM employees
WHERE id=?",
            handler, 103);
         //Display values
         System.out.print("Result: " + Arrays.toString(result));
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
Connecting to database...
Result: [103, 33, Sumit, Mittal]
```

11. Apache Commons DBUtils — BeanHandler Class

The **org.apache.commons.dbutils.BeanHandler** is the implementation of ResultSetHandler interface and is responsible to convert the first ResultSet row into a JavaBean. This class is thread safe.

Class Declaration

Following is the declaration for org.apache.commons.dbutils.BeanHandler class:

```
public class BeanHandler<T>
   extends Object implements ResultSetHandler<T>
```

Usage

- **Step 1** – Create a connection object.
- **Step 2** – Get implementation of ResultSetHandler as BeanHandler object.
- **Step 3** – Pass resultSetHandler to QueryRunner object, and make database operations.

Example

Following example will demonstrate how to read a record using BeanHandler class. We will read one of the available record in Employees Table and map it to the Employee bean.

Syntax

The syntax is given below:

```
Employee emp = queryRunner.query(conn, "SELECT * FROM employees WHERE first=?",
resultHandler, "Sumit");
```

Where,

- **resultHandler** – BeanHandler object to map result set to Employee object.
- **queryRunner** – QueryRunner object to read employee object from database.

To understand the above-mentioned concepts related to DBUtils, let us write an example which will run a read query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.

2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.ResultSetHandler;
import org.apache.commons.dbutils.handlers.BeanHandler;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";

   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();

      //Step 1: Register JDBC driver
      DbUtils.loadDriver(JDBC_DRIVER);

      //Step 2: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);

      //Step 3: Create a ResultSet Handler to handle Employee Beans
      ResultSetHandler<Employee> resultHandler
         = new BeanHandler<Employee>(Employee.class);

      try {
         Employee emp = queryRunner.query(conn, "SELECT * FROM employees WHERE

first=?",
```

```
                resultHandler, "Sumit");
         //Display values
         System.out.print("ID: " + emp.getId());
         System.out.print(", Age: " + emp.getAge());
         System.out.print(", First: " + emp.getFirst());
         System.out.println(", Last: " + emp.getLast());
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message.

```
ID: 103, Age: 28, First: Sumit, Last: Mittal
```

12. Apache Commons DBUtils — BeanListHandler Class

The **org.apache.commons.dbutils.BeanListHandler** is the implementation of ResultSetHandler interface and is responsible to convert the ResultSet rows into list of Java Bean. This class is thread safe.

Class Declaration

Following is the declaration for org.apache.commons.dbutils.BeanListHandler class:

```
public class BeanListHandler<T>
   extends Object implements ResultSetHandler<List<T>>
```

Usage

- **Step 1** – Create a connection object.
- **Step 2** – Get implementation of ResultSetHandler as BeanListHandler object.
- **Step 3** – Pass resultSetHandler to QueryRunner object, and make database operations.

Example

Following example will demonstrate how to read a list of records using BeanListHandler class. We will read available records in Employees Table and map them to list of Employee beans.

Syntax

The syntax is given below:

```
List<Employee> empList = queryRunner.query(conn, "SELECT * FROM employees",
resultHandler);
```

Where,

- **resultHandler** – BeanListHandler object to map the result sets to the list of Employee objects.
- **queryRunner** – QueryRunner object to read employee object from database.

To understand the above-mentioned concepts related to DBUtils, let us write an example which will run a read query. To write our example, let us create a sample application.

Step	Description
1	Update the file *MainApp.java* created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.
2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
```

```
      this.last = last;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.List;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.ResultSetHandler;
import org.apache.commons.dbutils.handlers.BeanListHandler;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";

   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();

      //Step 1: Register JDBC driver
      DbUtils.loadDriver(JDBC_DRIVER);

      //Step 2: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);

      //Step 3: Create a ResultSet Handler to handle List of Employee Beans
      ResultSetHandler<List<Employee>> resultHandler = new
```

```
BeanListHandler<Employee>(Employee.class);


      try {
         List<Employee> empList = queryRunner.query(conn, "SELECT * FROM
employees", resultHandler);
         for(Employee emp: empList ) {


            //Display values
            System.out.print("ID: " + emp.getId());
            System.out.print(", Age: " + emp.getAge());
            System.out.print(", First: " + emp.getFirst());
            System.out.println(", Last: " + emp.getLast());
         }
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
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: 28, First: Sumit, Last: Mittal
```

13. Apache Commons DBUtils — ArrayListHandler Class

The **org.apache.commons.dbutils.ArrayListHandler** is the implementation of ResultSetHandler interface and is responsible to convert the ResultSet rows into an object[]. This class is thread safe.

Class Declaration

Following is the declaration for org.apache.commons.dbutils.ArrayListHandler class:

```
public class ArrayListHandler
   extends AbstractListHandler<Object[]>
```

Usage

- **Step 1** – Create a connection object.
- **Step 2** – Get implementation of ResultSetHandler as ArrayListHandler object.
- **Step 3** – Pass resultSetHandler to QueryRunner object, and make database operations.

Example

Following example will demonstrate how to read a list of records using ArrayListHandler class. We will read available records in Employees Table as object[].

Syntax

The syntax is as follows:

```
List<Object> result = queryRunner.query(conn, "SELECT * FROM employees", new
ArrayListHandler());
```

Where,

- **resultHandler** – ArrayListHandler object to map result sets to list of object[].
- **queryRunner** – QueryRunner object to read employee object from database.

To understand the above-mentioned concepts related to DBUtils, let us write an example which will run a read query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.

2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Arrays;
import java.util.List;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.handlers.ArrayListHandler;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";

   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();

      //Step 1: Register JDBC driver
      DbUtils.loadDriver(JDBC_DRIVER);

      //Step 2: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);

      try {
         List<Object[]> result = queryRunner.query(conn, "SELECT * FROM
employees"
            , new ArrayListHandler());
         for(Object[] objects : result) {

            System.out.println(Arrays.toString(objects));
```

```
         }
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
[100, 18, Zara, Ali]
[101, 25, Mahnaz, Fatma]
[102, 30, Zaid, Khan]
[103, 28, Sumit, Mittal]
```

14. Apache Commons DBUtils — MapListHandler Class

The **org.apache.commons.dbutils.MapListHandler** is the implementation of ResultSetHandler interface and is responsible to convert the ResultSet rows into list of Maps. This class is thread safe.

Class Declaration

Following is the declaration for org.apache.commons.dbutils.MapListHandler class:

```
public class MapListHandler
   extends AbstractListHandler<Map<String,Object>>
```

Usage

- **Step 1** – Create a connection object.
- **Step 2** – Get implementation of ResultSetHandler as MapListHandler object.
- **Step 3** – Pass resultSetHandler to QueryRunner object, and make database operations.

Example

Following example will demonstrate how to read a list of records using MapListHandler class. We will read available records in Employees Table as list of maps.

Syntax

The syntax is given below:

```
List<Map<String, Object>> result = queryRunner.query(conn, "SELECT * FROM
employees", new MapListHandler());
```

Where,

- **resultHandler** – MapListHandler object to map result sets to list of maps.
- **queryRunner** – QueryRunner object to read employee object from database.

To understand the above-mentioned concepts related to DBUtils, let us write an example which will run a read query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.

2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.List;
import java.util.Map;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.handlers.MapListHandler;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";

   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();

      //Step 1: Register JDBC driver
      DbUtils.loadDriver(JDBC_DRIVER);

      //Step 2: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL, USER, PASS);

      try {
         List<Map<String, Object>> result
            = queryRunner.query(conn, "SELECT * FROM employees", new
MapListHandler());
         System.out.println(result);

      } finally {
```

```
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
Connecting to database...
[{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=33, first=Sumit, last=Mittal}]
```

15. Apache Commons DBUtils — Custom Handler

We can create our own custom handler by implementing ResultSetHandler interface or by extending any of the existing implementation of ResultSetHandler.

In the example given below, we have created a Custom Handler, EmployeeHandler by extending BeanHandler class.

To understand the above mentioned concepts related to DBUtils, let us write an example which will run a read query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.
2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   private String name;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
```

```
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
}
```

Following is the content of the **EmployeeHandler.java** file.

```
import java.sql.ResultSet;
import java.sql.SQLException;

import org.apache.commons.dbutils.handlers.BeanHandler;

public class EmployeeHandler extends BeanHandler<Employee> {

   public EmployeeHandler() {
      super(Employee.class);
   }

   @Override
   public Employee handle(ResultSet rs) throws SQLException {
      Employee employee = super.handle(rs);
      employee.setName(employee.getFirst() +", " + employee.getLast());
      return employee;
```

```
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.ResultSetHandler;
import org.apache.commons.dbutils.handlers.BeanHandler;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";

   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();
      DbUtils.loadDriver(JDBC_DRIVER);
      conn = DriverManager.getConnection(DB_URL, USER, PASS);
      EmployeeHandler employeeHandler = new EmployeeHandler();

      try {
         Employee emp = queryRunner.query(conn, "SELECT * FROM employees WHERE
first=?",
            employeeHandler, "Sumit");

         //Display values
         System.out.print("ID: " + emp.getId());
```

```
         System.out.print(", Age: " + emp.getAge());

         System.out.print(", Name: " + emp.getName());
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
ID: 103, Age: 28, Name: Sumit, Mittal
```

16. Apache Commons DBUtils — Custom Row Processor

In case, column names in a database table and equivalent javabean object names are not similar, then we can map them by using customised BasicRowProcessor object. See the example given below.

To understand the above mentioned concepts related to DBUtils, let us write an example which will run a read query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.
2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   private String name;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
```

```
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
}
```

Following is the content of the **EmployeeHandler.java** file.

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.Map;

import org.apache.commons.dbutils.handlers.BeanHandler;
import org.apache.commons.dbutils.BeanProcessor;
import org.apache.commons.dbutils.BasicRowProcessor;

public class EmployeeHandler extends BeanHandler<Employee> {

   public EmployeeHandler() {
      super(Employee.class, new BasicRowProcessor(new
BeanProcessor(mapColumnsToFields())));
   }

   @Override
```

```
   public Employee handle(ResultSet rs) throws SQLException {
      Employee employee = super.handle(rs);
      employee.setName(employee.getFirst() +", " + employee.getLast());
      return employee;
   }

   public static Map<String, String> mapColumnsToFields() {
      Map<String, String> columnsToFieldsMap = new HashMap<>();
      columnsToFieldsMap.put("ID", "id");
      columnsToFieldsMap.put("AGE", "age");
      return columnsToFieldsMap;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;

public class MainApp {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";

   public static void main(String[] args) throws SQLException {
      Connection conn = null;
      QueryRunner queryRunner = new QueryRunner();
      DbUtils.loadDriver(JDBC_DRIVER);
      conn = DriverManager.getConnection(DB_URL, USER, PASS);
```

```
      EmployeeHandler employeeHandler = new EmployeeHandler();


      try {
         Employee emp = queryRunner.query(conn, "SELECT * FROM employees WHERE
first=?",
           employeeHandler, "Sumit");


         //Display values
         System.out.print("ID: " + emp.getId());
         System.out.print(", Age: " + emp.getAge());
         System.out.print(", Name: " + emp.getName());
      } finally {
         DbUtils.close(conn);
      }
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
ID: 103, Age: 28, Name: Sumit, Mittal
```

17. Apache Commons DBUtils — Using DataSource

So far, we have used connection object, while using QueryRunner. We can also use datasource seamlessly. The following example will demonstrate, how to read a record using Read query with the help of QueryRunner and datasource. We will read a record from Employees Table.

Syntax

The syntax is as follows:

```
QueryRunner queryRunner = new QueryRunner( dataSource );

Employee emp = queryRunner.query("SELECT * FROM employees WHERE first=?",
resultHandler, "Sumit");
```

Where,

- **dataSource** – DataSource object configured.
- **resultHandler** – ResultSetHandler object to map the result set to Employee object.
- **queryRunner** – QueryRunner object to read an employee object from database.

To understand the above mentioned concepts related to DBUtils, let us write an example, which will run a read query. To write our example, let us create a sample application.

Step	Description
1	Update the file **MainApp.java** created under chapter **DBUtils - First Application,** which is available at https://www.tutorialspoint.com/dbutils/dbutils_first_application.htm.
2	Compile and run the application as explained below.

Following is the content of the **Employee.java**.

```
public class Employee {
   private int id;
   private int age;
   private String first;
   private String last;
   public int getId() {
      return id;
```

```
   }
   public void setId(int id) {
      this.id = id;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getFirst() {
      return first;
   }
   public void setFirst(String first) {
      this.first = first;
   }
   public String getLast() {
      return last;
   }
   public void setLast(String last) {
      this.last = last;
   }
}
```

Following is the content of the **CustomDatasource.java**.

```
import javax.sql.DataSource;
import org.apache.commons.dbcp2.BasicDataSource;

public class CustomDataSource {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost:3306/emp";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "admin";
   private static DataSource datasource;
```

```
   private static final BasicDataSource basicDataSource;

   static {
      basicDataSource = new BasicDataSource();
      basicDataSource.setDriverClassName(JDBC_DRIVER);
      basicDataSource.setUsername(USER);
      basicDataSource.setPassword(PASS);
      basicDataSource.setUrl(DB_URL);
   }

   public static DataSource getInstance() {
      return basicDataSource;
   }
}
```

Following is the content of the **MainApp.java** file.

```
import java.sql.SQLException;

import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.ResultSetHandler;
import org.apache.commons.dbutils.handlers.BeanHandler;

public class MainApp {
   public static void main(String[] args) throws SQLException {

      DbUtils.loadDriver(JDBC_DRIVER);
      QueryRunner run = new QueryRunner(CustomDataSource.getInstance());
      ResultSetHandler<Employee> resultHandler = new
BeanHandler<Employee>(Employee.class);

      Employee emp = queryRunner.query("SELECT * FROM employees WHERE id=?",
         resultHandler, 103);

      //Display values
      System.out.print("ID: " + emp.getId());
      System.out.print(", Age: " + emp.getAge());
```

```
      System.out.print(", First: " + emp.getFirst());

      System.out.println(", Last: " + emp.getLast());
   }
}
```

Once you are done creating the source files, let us run the application. If everything is fine with your application, it will print the following message:

```
ID: 103, Age: 33, First: Sumit, Last: Mittal
```