To execute an SQL statement using JDBC in Java, you can follow these steps:

Step 1: Create a Connection

Establish a connection with the database as described in the previous steps. Make sure you have the Connection object available for executing the SQL statement.

```
Connection connection = DriverManager.getConnection(url, username,
password);
```

Step 2: Create a Statement

Create a Statement object using the Connection object. The Statement interface allows you to execute SQL statements.

```
Statement statement = connection.createStatement();
```

Step 3: Execute the SQL Statement

Use the execute, executeQuery, or executeUpdate methods of the Statement object to execute the SQL statement based on your needs:

- execute: Use this method if your SQL statement can return multiple result sets or does not return any result (such as creating tables or executing stored procedures). It returns a boolean value indicating whether the first result is a ResultSet or not.
- executeQuery: Use this method if your SQL statement returns a ResultSet, which

represents the data retrieved from the database. It returns a ResultSet object.

- executeUpdate: Use this method if your SQL statement is an update statement (such as INSERT, UPDATE, DELETE), which modifies the database. It returns an int value representing the number of rows affected.

Here are examples of each method:

```
// Execute a statement that does not return a result (e.g., CREATE
TABLE)
boolean hasResultSet = statement.execute("CREATE TABLE mytable (id
INT, name VARCHAR(50))");

// Execute a query statement that returns a result set
ResultSet resultSet = statement.executeQuery("SELECT * FROM mytable");

// Execute an update statement that modifies the database
int rowsAffected = statement.executeUpdate("UPDATE mytable SET name =
'John' WHERE id = 1");
```

Step 4: Process the Result (if applicable)

If your SQL statement returns a ResultSet, you can process the retrieved data using the ResultSet object. Iterate over the result set using methods like next, and retrieve data using methods like getString, getInt, etc.

```
while (resultSet.next()) {
    String name = resultSet.getString("name");
    int age = resultSet.getInt("age");
    System.out.println("Name: " + name + ", Age: " + age);
```

```
}
```

Step 5: Close the Resources

As mentioned before, it's important to close the resources (ResultSet, Statement, and Connection) after you finish using them to release any held resources:

```
resultSet.close();
statement.close();
connection.close();
```

Make sure to handle any exceptions that may occur during the execution of SQL statements using try-catch blocks or propagating the exceptions to the calling code.

That's it! You now know how to execute SQL statements using JDBC in Java. Remember to handle exceptions appropriately and close the resources to ensure the efficient usage of database connections.

Related posts:

1. Can Java have same name variable
2. Types of variables in Java programming
3. JAVA and its Support Systems
4. JAVA environment
5. JAVA program structure
6. Tokens
7. Java statements

8. Java virtual machine
9. C++ Versus JAVA
10. Constants and Variables in Java
11. Data types JAVA
12. Defining a class
13. Constructor in JAVA
14. Array in Java
15. Applet
16. Applets Vs Applications
17. Writing applets
18. Applets life cycle
19. Creating an Executable Applet
20. Graphics in Applet
21. Applet image display
22. Applet digital clock
23. Applet mouse event handling
24. JDBC
25. Process the result
26. CLOSE THE DATABASE CONNECTION
27. File handling
28. Define a class to declare an integer array of size n and accept the elements into the array.
29. Define a class to declare an array of size 20 of the double datatype, accept the elements into the array and perform the following: Calculate and print the sum of all the elements.
30. Java program for String, to uppercase, to equal, length of string
31. Write a Java program for Buble sort.

32. Write a Java program String to uppercase and count words startig with ‘A’
33. How to set path in Java
34. Understanding public static void main (String args[]){ } in Java
35. Difference between static and non static methods in Java