

Calling MySQL Stored Procedures from JDBC

In this tutorial, you will learn how to call MySQL stored procedures from JDBC using CallableStatement object.

Before you start

For the sake of demonstration, we will create a new stored procedure named `get_candidate_skill` that accepts `candidate_id` as the IN parameter and returns a result set that contains the skills of the candidate.

```
1 DELIMITER $$
2 CREATE PROCEDURE get_candidate_skill(IN candidate
3   _id INT)
4 BEGIN
5     SELECT candidates.id, first_name, last_name, skill
6     ls.name AS skill
7     FROM candidates
8     INNER JOIN candidate_skills ON candidates.id = c
9     andidate_skills.candidate_id
10    INNER JOIN skills ON skills.id = candidate_skill
11    s.skill_id
12    WHERE candidates.id = candidate_id;
13     END$$
14 DELIMITER ;
```

Let's call this stored procedure for candidate id with value 122.

```
1 CALL get_candidate_skill(122);
```

	id	first_name	last_name	skill
▶	122	John	Doe	Java
	122	John	Doe	JDBC
	122	John	Doe	MySQL

Introducing CallableStatement and stored procedure call syntax

To call stored procedures or stored functions in MySQL from JDBC, you use `CallableStatement` object, which inherits from `PreparedStatement` object. The general syntax of calling a stored procedure is as follows:

```
1 {?= call procedure_name(param1,param2,...)}
```

You wrap the stored procedure call within braces (`{}`). If the stored procedure returns a value, you need to add the question mark and equal (`?=`) before the `call` keyword. If a stored procedure does not return any values, you just omit the `?=` sign. In case the stored procedure accepts any parameters, you list them within the opening and closing parentheses after the stored procedure's name.

The following are examples of using the syntax for calling stored procedures in different contexts:

Syntax	Stores Procedures
<code>{ call procedure_name() }</code>	Accept no parameters and return no value
<code>{ call procedure_name(?,?) }</code>	Accept two parameters and return no value
<code>{?= call procedure_name() }</code>	Accept no parameter and return value
<code>{?= call procedure_name(?) }</code>	Accept one parameter and return value

Notice that question mark placeholder (?) can be used for both IN ,OUT, and INOUT parameters. For detailed information on different parameter types in stored procedures, check it out [MySQL stored procedure parameters tutorial](#).

JDBC MySQL stored procedure example

First, open a connection to MySQL server by creating a newConnection object.

```
1 Connection conn = DriverManager.getConnection();
```

Then, prepare a stored procedure call and create aCallableStatement object by calling prepareCall() method of the Connection object.

```
1 String query = "{CALL get_candidate_skill(?)}";
2 CallableStatement stmt = conn.prepareCall(query)
```

Next, pass all the parameters to the stored procedure. In this case, the get_candidate_skill stored procedure accepts only one IN parameter.

```
1 stmt.setInt(1, candidateId);
```

After that, execute the stored procedure by calling theexecuteQuery() method of the CallableStatement object. It returns a result set in this case.

```
1 ResultSet rs = stmt.executeQuery();
```

Finally, traverse the ResultSet to display the results.

```
1 while (rs.next()) {
2     System.out.println(String.format("%s - %s",
3                                     rs.getString("first_name") +
4     " "
5                                     + rs.getString("last_name"),
6                                     rs.getString("skill")));
}
```

The following is the complete example of calling the MySQL stored procedure from JDBC.

```
1 package org.mysqltutorial;
```

```

2
3 import java.sql.Connection;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.sql.CallableStatement;
7
8 /**
9  *
10  * @author mysqltutorial.org
11  */
12 public class Main {
13
14     /**
15      * Get skills by candidate id
16      *
17      * @param candidateId
18      */
19     public static void getSkills(int candidateId)
20     {
21         //
22         String query = "{ call get_candidate_ski
23 l(?) }";
24         ResultSet rs;
25
26         try (Connection conn = MySQLJDBCUtil.getC
27 onnection());
28             CallableStatement stmt = conn.pre
29 pareCall(query)) {
30
31             stmt.setInt(1, candidateId);
32
33             rs = stmt.executeQuery();
34             while (rs.next()) {
35                 System.out.println(String.format("
36 - %s",
37 rs.getString("first_name")
38 + " "
39 + rs.getString("last_nam
40 e"),

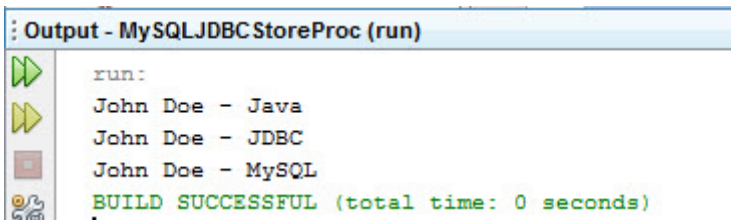
```

```

41         rs.getString("skill")));
42     }
43     } catch (SQLException ex) {
44         System.out.println(ex.getMessage());
45     }
46 }
47
48 /**
49  *
50  * @param args
51  */
52 public static void main(String[] args) {
53     getSkills(122);
54 }
55 }

```

Let's run the program.



```

Output - MySQLJDBCStoreProc (run)
run:
John Doe - Java
John Doe - JDBC
John Doe - MySQL
BUILD SUCCESSFUL (total time: 0 seconds)

```

The program works as expected.

In this tutorial, we have shown you how to call a stored procedure in MySQL database from a Java program using JDBC CallableStatement object.