Updating Data in MySQL Using JDBC PreparedStatement

?

In this tutorial, you will learn how to update data in MySQL database using JDBC PreparedStatement interface.

The `PreparedStatement` interface extends the `Statement`interface that provides some more advanced features as follows:

- Add parameters into you SQL statement using placeholders in the form of question marks (?). This helps you avoid SQL injection.
- Reuse the `PreparedStatement` with new parameters in case you need to execute the SQL statement multiple times with different parameters.
- Help increase performance of the executed statement by precompiling the SQL statement.

We will use the `PreparedStatement` to update last names of candidates in the `candidates` table.

First, you open a connection to MySQL database by reusing the utility class named `MySQLJDBCUtil` that we developed in theprevious tutorial.

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

Second, you construct a SQL UPDATE statement and create`PreparedStatement` object by calling the`prepareStatement()` method of the `Connection` object. The`prepareStatement()` method accepts various parameters. In this example, you pass in a string which is a SQL statement.

```
String sqlUpdate = "UPDATE candidates "
                 + "SET last_name = ? "
                 + "WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(sqlUpdate);
```

Notice that there are two question marks (?) as the placeholders for `last_name` and `id` fields.

Third, you supply values for the placeholders one-by-one by using `setYYY()` method of the `PreparedStatement` interface where YYY is the data type of the placeholder. For example, you want to update last name of candidate with id `100` to `William`, you can set the values for the placeholders as follows:

```
String lastName = "William";
int id = 100;

pstmt.setString(1, lastName);
pstmt.setInt(2, id);
```

Fourth, you send the `UPDATE` statement with the values for the placeholders to MySQL by calling `executeUpdate()` method of the `PreparedStatement` interface. This method takes no arguments and returns the number of row affected.

```
int rowAffected = pstmt.executeUpdate();
```

In case you want to reuse the `PreparedStatement` , you need to populate new values for the placeholders and call the method`executeUpdate()` again. For example, if you want to update the last name of candidate with id `101` to `Grohe` , you can do it as follows:

```
 // reuse the prepared statement
lastName = "Grohe";
id = 101;

pstmt.setString(1, lastName);
pstmt.setInt(2, id);

rowAffected = pstmt.executeUpdate();
```

As always, you should close the `PreparedStatement` by calling its `close()` method.

```
pstmt.close()
```

In case you use the try-with-resources statement, you don't have to explicitly do this. The following illustrates the complete example of using `PreparedStatement` to update data.

```
package org.mysqltutorial;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

/**
 *
 * @author mysqltutorial.org
 */
public class Main {

    /**
     * Update candidate demo
     */
    public void update() {

        String sqlUpdate = "UPDATE candidates "
                + "SET last_name = ? "
                + "WHERE id = ?";

        try (Connection conn = MySQLJDBCUtil.getC
onnection();
                PreparedStatement pstmt = conn.pr
epareStatement(sqlUpdate)) {

            // prepare data for update
            String lastName = "William";
            int id = 100;
            pstmt.setString(1, lastName);
            pstmt.setInt(2, id);

```

```
            int rowAffected = pstmt.executeUpdate(
;
            System.out.println(String.format("Row
 affected %d", rowAffected));

            // reuse the prepared statement
            lastName = "Grohe";
            id = 101;
            pstmt.setString(1, lastName);
            pstmt.setInt(2, id);

            rowAffected = pstmt.executeUpdate();
            System.out.println(String.format("Row
 affected %d", rowAffected));

        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }

    /**
     * main method
     *
     * @param args
     */
    public static void main(String[] args) {
        update();


    }
}
```

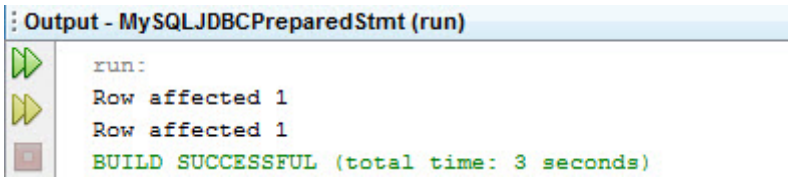
Before executing the program, let's examine the current candidates with id 100 and 101 in the mysqljdbc database:

```
SELECT * FROM candidates
WHERE id = 100 OR id = 101;
```

The result of the query is as follows:

id	first_name	last_name	dob	phone	email
100	Alexander	France	1980-04-09	(342) 555-1760	alexander.f@yahoo.com
101	Dan	Loye	1980-04-10	(203) 555-4407	dan.l@yahoo.com

By executing the Java program above, you will see the following output:



If you execute the SELECT statement again, you will see that the changes have been applied to the candidates table.

id	first_name	last_name	dob	phone	email
100	Alexander	William	1980-04-09	(342) 555-1760	alexander.f@yahoo.com
101	Dan	Grohe	1980-04-10	(203) 555-4407	dan.l@yahoo.com

It is important to note that you can use any statement such asSELECT, INSERT, DELETE, etc with `PreparedStatement` interface.

In this tutorial, we have shown you how to update data in MySQL using JDBC PrepareStatement interface.