Lars Vogel (c) 2009, 2016 vogella GmbH Version 1.4, 29.09.2016

MySQL and Java JDBC. This tutorial describes how to use Java JDBC to connect to MySQL and perform SQL queries, database inserts and deletes.

1. Connection to database with Java

The interface for accessing relational databases from Java is*Java Database Connectivity (JDBC)*. Via JDBC you create a connection to the database, issue database queries and update as well as receive the results.

JDBC provides an interface which allows you to perform SQL operations independently of the instance of the used database. To use JDBC, you require the database specific implementation of the JDBC driver.

2. Introduction to MySQL

To learn to install and use MySQL please seeMySQL - Tutorial.

The following description will assume that you have successfully installed MySQL and know how to access MySQL via the command line.

3. MySQL JDBC driver

To connect to MySQL from Java, you have to use the JDBC driver from MySQL. The MySQL JDBC driver is called *MySQL Connector/J*. You find the latest MySQL JDBC driver under the following URL: http://dev.mysql.com/downloads/connector/j.

The download contains a `JAR` file which we require later.

4. Exercise: create example database

In this exercise you create a new database, a new user and an example table. For this connect to the MySQL server via the `mysql` command line client.

Create a new database called *feedback* and start using it with the following command.

```
create database feedback;
use feedback;
```

Create a user with the following command.

```
CREATE USER sqluser IDENTIFIED BY 'sqluserpw';

grant usage on *.* to sqluser@localhost identified by 'sqluserpw';
grant all privileges on feedback.* to sqluser@localhost;
```

Now create a sample database table with example content via the following SQL statement.

```
CREATE TABLE comments (
        id INT NOT NULL AUTO_INCREMENT,
        MYUSER VARCHAR(30) NOT NULL,
        EMAIL VARCHAR(30),
        WEBPAGE VARCHAR(100) NOT NULL,
        DATUM DATE NOT NULL,
        SUMMARY VARCHAR(40) NOT NULL,
        COMMENTS VARCHAR(400) NOT NULL,
        PRIMARY KEY (ID)
    );

INSERT INTO comments values (default, 'lars', 'myemail@gmail.com','https://www.vogella.com/', '2009-09-14
10:33:11', 'Summary','My first comment' );
```

5. Java JDBC

Create a Java project and a package called *de.vogella.mysql.first*.

Create a `lib` folder and copy the JDBC driver into this folder. Add the JDBC driver to your classpath. See Adding jars to the classpath for details.

Create the following class to connect to the MySQL database and perform queries, inserts and deletes. It also prints the metadata (table name, column names) of a query result.

```
package de.vogella.mysql.first;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Date;

public class MySQLAccess {
    private Connection connect = null;
    private Statement statement = null;
    private PreparedStatement preparedStatement = null;
    private ResultSet resultSet = null;

    public void readDataBase() throws Exception {
        try {
            // This will load the MySQL driver, each DB has its own driver
            Class.forName("com.mysql.jdbc.Driver");
            // Setup the connection with the DB
            connect = DriverManager
                    .getConnection("jdbc:mysql://localhost/feedback?"
                            + "user=sqluser&password=sqluserpw");

            // Statements allow to issue SQL queries to the database
            statement = connect.createStatement();
            // Result set get the result of the SQL query
            resultSet = statement
                    .executeQuery("select * from feedback.comments");
            writeResultSet(resultSet);
```

```
            // PreparedStatements can use variables and are more efficient
            preparedStatement = connect
                    .prepareStatement("insert into  feedback.comments values (default, ?, ?, ?, ? , ?, ?)");
            // "myuser, webpage, datum, summary, COMMENTS from feedback.comments");
            // Parameters start with 1
            preparedStatement.setString(1, "Test");
            preparedStatement.setString(2, "TestEmail");
            preparedStatement.setString(3, "TestWebpage");
            preparedStatement.setDate(4, new java.sql.Date(2009, 12, 11));
            preparedStatement.setString(5, "TestSummary");
            preparedStatement.setString(6, "TestComment");
            preparedStatement.executeUpdate();

            preparedStatement = connect
                    .prepareStatement("SELECT myuser, webpage, datum, summary, COMMENTS from feedback.comments");
            resultSet = preparedStatement.executeQuery();
            writeResultSet(resultSet);

            // Remove again the insert comment
            preparedStatement = connect
            .prepareStatement("delete from feedback.comments where myuser= ? ; ");
            preparedStatement.setString(1, "Test");
            preparedStatement.executeUpdate();

            resultSet = statement
            .executeQuery("select * from feedback.comments");
            writeMetaData(resultSet);

        } catch (Exception e) {
            throw e;
        } finally {
            close();
        }

    }

    private void writeMetaData(ResultSet resultSet) throws SQLException {
        //  Now get some metadata from the database
        // Result set get the result of the SQL query

        System.out.println("The columns in the table are: ");

        System.out.println("Table: " + resultSet.getMetaData().getTableName(1));
        for  (int i = 1; i<= resultSet.getMetaData().getColumnCount(); i++){
            System.out.println("Column " +i  + " "+ resultSet.getMetaData().getColumnName(i));
        }
    }

    private void writeResultSet(ResultSet resultSet) throws SQLException {
        // ResultSet is initially before the first data set
        while (resultSet.next()) {
            // It is possible to get the columns via name
            // also possible to get the columns via the column number
            // which starts at 1
            // e.g. resultSet.getSTring(2);
            String user = resultSet.getString("myuser");
            String website = resultSet.getString("webpage");
            String summary = resultSet.getString("summary");
            Date date = resultSet.getDate("datum");
            String comment = resultSet.getString("comments");
            System.out.println("User: " + user);
            System.out.println("Website: " + website);
            System.out.println("summary: " + summary);
            System.out.println("Date: " + date);
            System.out.println("Comment: " + comment);
        }
    }

    // You need to close the resultSet
    private void close() {
        try {
            if (resultSet != null) {
                resultSet.close();
            }

            if (statement != null) {
                statement.close();
            }

            if (connect != null) {
```

```
                connect.close();
            }
        } catch (Exception e) {

        }
    }

}
```

Create the following main program to test your class.

```
package de.vogella.mysql.first.test;

import de.vogella.mysql.first.MySQLAccess;

public class Main {
    public static void main(String[] args) throws Exception {
        MySQLAccess dao = new MySQLAccess();
        dao.readDataBase();
    }

}
```

6. Links and Literature

MySQL homepage

Download link for MySQL

7. vogella training and consulting support

Appendix A: Copyright, License and Source code

Licence

Source code

Support free tutorials

Version unspecified
Last updated 2019-05-23 14:17:25 +0200