MySQL and Java

Ömer Erdem Demir

January 25, 2006

1 Requirements

You will need:

1. `java` and `javac`
2. MySQL installed. Directions for installing MySQL on CSIF machines can be found at http://csifdocs.cs.ucdavis.edu/tiki-index.php?page=CSIF+MySQL+4.x+Install
3. MySQL JDBC driver. You can download it from http://dev.mysql.com/downloads/connector/j/3.1.html
   Extract `mysql-connector-java-3.1.12-bin.jar` (or the latest version you have) from the Connector/J archive to your home directory, you will not need the other files.

2 Setting up the tutorial database

In this section we will create a new database, a new user, and a very simple table. MySQL has a two level directory like hierarchy for keeping databases and tables. At the root there is MySQL; under root you can only create "databases." Database is almost like a directory, you can create "tables" under a database. Follow the steps listed below.

1. Start the mysql server (follow the CSIF MySQL tutorial).
2. Check if mysql server is running.

```
$ mysqladmin -u root -p status
Uptime:  434 Threads:  1 Questions:  86 Slow queries:  0 ...
```

3. Start the mysql client. We will use the command line client to create a new database, a new user and a table in the new database.

(a)
```
$ mysql -u root -p
Welcome to the MySQL monitor.  Commands end with ; or \g.
⋮
mysql>
```

(b) Create a new database named `ecs160tutorial`.

```
mysql> CREATE DATABASE ecs160tutorial;
Query OK, 1 row affected (0.06 sec)
```

(c) Create a user with all privileges on this database. The user name will be `tutorialuser` and the password will be `123456`. Although this is NOT good practice, it will suffice.

```
mysql> GRANT ALL ON ecs160tutorial.* TO tutorialuser@'%'
    -> IDENTIFIED BY  '123456';
mysql> GRANT ALL ON ecs160tutorial.* TO tutorialuser@'localhost'
    -> IDENTIFIED BY  '123456';
```

(d) Quit mysql client. We'll reconnect as `tutorialuser` to the `ecs160tutorial` database to setup a table.

```
mysql> quit
Bye
$ mysql -u tutorialuser -p ecs160tutorial
Enter password:
Welcome...
⋮
mysql>
```

(e) Now we will create a simple table with two columns, name and last name.

```
mysql> CREATE TABLE simple_table (name CHAR(128), last_name CHAR(128));
Query OK, 0 rows affected (0.01 sec)
```

`show tables` command will list all the tables created in this database.

```
mysql> show tables;
+--------------------------+
| Tables_in_ecs160tutorial |
+--------------------------+
| simple_table             |
+--------------------------+
1 row in set (0.00 sec)
```

So far we set up a new database for this tutorial, created a user and a very simple table. I think it is a good idea to create a new database and user for your project as we did in this tutorial. In the next section, I'll describe how to connect to the `ecs160tutorial` database from a `Java` program and execute simple queries.

3 Connecting to a MySQL database from a Java program using the Connector/J JDBC driver

I assume that you downloaded and installed Connector/J. If you haven't done so, read section 1 for the requirements.

You can connect to the MySQL database in two steps. Those steps are detailed below.

1. First load the driver.

```
Class driver_class=null;
try {
   driver_class = Class.forName("com.mysql.jdbc.Driver");
} catch (ClassNotFoundException e) {
   e.printStackTrace();
}
System.out.println("Found driver " + driver_class);
```

   We don't need to register the driver, once it is loaded it will be used for connection requests to mysql databases.

2. Next step is to connect to the MySQL server and the ecs160tutorial database. Recall that the user name is tutorialuser and the password is 123456.

```
Connection connection=null;
try {
        connection = DriverManager.getConnection
         ("jdbc:mysql://localhost:3306/ecs160tutorial","tutorialuser","123456");
} catch (SQLException e) {
        e.printStackTrace();
}

try {
        System.out.println
         ("Established connection to "+ connection.getMetaData().getURL());
} catch (SQLException e1) {
        e1.printStackTrace();
}
```

   You must have noticed that DriverManager.getConnection takes three arguments. The first argument is the URL of the server; URLs always start with jdbc:mysql:// and followed by the server address and the database name. Therefore, if you are running the MySQL server on a different machine you should replace localhost with the correct machine address, either name or IP address. Moreover, you'll need to

replace `3306` with the number of the port your MySQL server is listening on. Next component of the URL is the database name. The second argument is the user name and the last one is the password.

Next, we will switch back to the `mysql` client to populate `simple_table`.

1. Connect to the database using the `mysql` client.

```
$ mysql -u tutorialuser -p ecs160tutorial
Enter password:
Welcome...
⋮
mysql>
```

2. Now we will insert two items into our `simple_table`.

```
mysql> INSERT INTO simple_table VALUES ("omer","demir");
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO simple_table VALUES ("kivilcim","dogerlioglu-demir");
Query OK, 1 row affected (0.00 sec)
```

3. Run the following query, the output you see should be similar to the output given below.

```
mysql> SELECT * from simple_table;
+----------+-------------------+
| name     | last_name         |
+----------+-------------------+
| omer     | demir             |
| kivilcim | dogerlioglu-demir |
+----------+-------------------+
2 rows in set (0.00 sec)
```

Now, we will execute the same `SELECT` query from our Java program.

1. We will use the `connection` to create an empty statement.

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

2. Execute the `SELECT` query.

```
statement.execute("SELECT * FROM  simple_table");
```

3. Get the result set of the query.

```
ResultSet resset = statement.getResultSet();
```

See http://java.sun.com/j2se/1.4.2/docs/api/java/sql/ResultSet.html for the API documentation.

4. We are ready to print the result of the query. The result set returned by the statement initially points before the first row, thus you must call `next` to advance to the first row. See the code snippet below.

```
System.out.println("Row Name Last_Name");
while(resset.next())
{
        System.out.print(resset.getRow());
        System.out.print(" " + resset.getString("name"));
        System.out.println(" " + resset.getString("last_name"));
}
resset.close();
```

A row of the result set is made up of columns. We know the column names and the types of the columns of `simple_table`; they are `name` and `last_name` and both are type string. Therefore, we will use `getString` (remember column type) method with the column names.

The output should be similar to the one below.

```
Row Name Last_Name
1 omer demir
2 kivilcim dogerlioglu-demir
```

4 Summary

In this tutorial I explained, using MySQL, how to create a database, a user, and a simple table. I also explained how to connect to a MySQL database from a Java program and execute queries. The Java program I used as the example can be found in the appendix. You can use `javac` to compile the program. Don't forget to change the host address and the port number. To run it, you will need to pass `-classpath` option:

```
java -classpath /home/<user_name>/mysql-connector-java-3.1.12-bin.jar:$CLASSPATH:. Main
```

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

public class Main {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		Class driver_class=null;
		try {
			driver_class = Class.forName("com.mysql.jdbc.Driver");
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
			return;
		}
		System.out.println("Found driver " + driver_class);

		Connection connection=null;
		try {
			connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/ecs160tutorial","tutorialuser","12
3456");
		} catch (SQLException e) {
			e.printStackTrace();
			return;
		}

		try {
			System.out.println("Established connection to "+
connection.getMetaData().getURL());
		} catch (SQLException e1) {
			e1.printStackTrace();
		}

		Statement statement=null;
		try {
			statement = connection.createStatement();
			statement.execute("SELECT * FROM  simple_table");
			ResultSet resset = statement.getResultSet();
			System.out.println("Row Name Last_Name");
			while(resset.next())
			{
				System.out.print(resset.getRow());
				System.out.print(" " + resset.getString("name"));
				System.out.println(" " + resset.getString("last_name"));
			}
			resset.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}
		finally{
			if (statement != null)
			{
				try {
					statement.close();
				} catch (SQLException e) {
					e.printStackTrace();
```

```
                                }
                        }
                        if (connection != null)
                        {
                                try {
                                        connection.close();
                                } catch (SQLException e) {
                                        e.printStackTrace();
                                }
                        }

                }


        }
}
```