For Ubuntu versions : 5.10, 6.06, 6.10, 7.04, 9.10

Introduction

This document explains how to set up Java applications to communicate with the MySQL Database. This can be used either in Java development, or applications which use JDBC such as OpenOffice.Org. See Using MySQL, JDBC and OpenOffice for more information.

This document was written originally for 5.10, but has been updated for 6.06. Changes required for 5.10 are shown in *italics*

It has also been checked against 6.10 (Edgy Eft) and 7.04 (Feisty Fawn), which should be installed as for 6.06

Installation

Install mysql client, server and the jdbc connector, either via synaptic or by using the following

```
sudo apt-get install mysql-server
sudo apt-get install mysql-client
sudo apt-get install libmysql-java
```

On 5.10 you may need select "No configuration" for postfix if it is installed.

Set up MySQL default password

Set up the password for the root user as 'root' or whatever you want. The last entry is the password

```
mysqladmin -u root password root
```

Check you can connect to mysql

```
mysql -u root -p
```

then enter the password you just set and press return again. In this case it would be "root".

Create SQL Databases and tables

Create databases and tables or whatever, so you have some data to work with. You can disconnect and use mysql-query-browser for this if you wish. For the sample code below just enter

```
create database emotherearth;
```

User creation and privileges

Create a user with access to that table. Replace the items in square brackets by the database name, and the chosen user and password. Don't type in the square brackets !

```
grant all privileges on [database].* to [user]@localhost identified by "[password]";
flush privileges;
```

Note: Ubuntu 5.10 requires localhost.localdomain

This is required because when you connect using "mysql -u root -p" it connects directly to the Sql Server. The "grant" line creates access via 127.0.0.1 (localhost). This is no longer an issue on 6.10.

You can test this by

```
mysql -u [user] -h 127.0.0.1 -p
```

This will work only for the user you have just "granted", not for root. Counter-intuitively, the 'default' server and localhost/127.0.0.1 aren't the same thing

Eclipse

For those who are using Eclipse, you will likely to have a 'Class Not Found' exception. To fix this, go to: Project, click Properties, select Java Build Path, and choose the Libraries tab. Then select 'Add External JARs', and find '/usr/share/java/mysql-connector-java.jar'.

Setting up the user to use JDBC

Add these two lines to /home/[user]/.bashrc. I am running Java 5 which doesn't require the current directory to be on the Classpath ; I *think* Java 2 does, but I'm not going back to check it In that case you may need to have $CLASSPATH:.:/user/share/java/mysql.jar

```
CLASSPATH=$CLASSPATH:/usr/share/java/mysql.jar
export CLASSPATH
```

Alternatively, you can set it for all users, by editing /etc/environment (use sudo - sudo vi /etc/environment)

```
CLASSPATH=".:/usr/share/java/mysql.jar"
```

Log out and Log in again. (If you only edited /home/[user]/.bashrc you don't need to log out/in, only execute in a terminal "$source .bashrc" in your home dir). Start up a terminal and type:

```
echo $CLASSPATH
```

It should print out something like ":/usr/share/java/mysql.jar"

Testing in Java

It should now work. Here is some typical code (clearing up removed for simplicity)

```
import java.sql.*;
import java.util.Properties;

public class DBDemo
{
  // The JDBC Connector Class.
  private static final String dbClassName = "com.mysql.jdbc.Driver";

  // Connection string. emotherearth is the database the program
  // is connecting to. You can include user and password after this
  // by adding (say) ?user=paulr&password=paulr. Not recommended!

  private static final String CONNECTION =
                          "jdbc:mysql://127.0.0.1/emotherearth";

  public static void main(String[] args) throws
                             ClassNotFoundException,SQLException
  {
    System.out.println(dbClassName);
    // Class.forName(xxx) loads the jdbc classes and
    // creates a drivermanager class factory
    Class.forName(dbClassName);

    // Properties for user and password. Here the user and password are both 'paulr'
    Properties p = new Properties();
    p.put("user","paulr");
    p.put("password","paulr");

    // Now try to connect
    Connection c = DriverManager.getConnection(CONNECTION,p);

    System.out.println("It works !");
    c.close();
    }
}
```