Tutorial: Accessing a MySql database in Java (Eclipse)

--D. Thiebaut (talk) 20:03, 2 March 2015 (EST)

Contents

To access a remote database server from your Java program, you need to make sure you have several things setup first:

1. A database server installed on a computer, with the port 3306 opened, so that computers on the outside can access the database running on it. You can bypass this requirement if you run the database server on your own computer, in which case the host will be *localhost*, instead of some string of the form *name.domain.com*.
2. A Java library, called a *connector* that will take SQL commands generated by your Java program, and will send them to the MySQL server over the network.
3. A database setup with one or several tables, all residing on the MySQL server. The best way to create tables on a MySQL server is to use a package such as **PhpMyAdmin**. It is free, and can be installed on any computer, as a Web application. See the PhpMyAdmin documentation for more information.

This tutorial concentrates on Step 2, above.

Steps

These steps are inspired for a good tutorial which can be found on www.vogella.com (http://www.vogella.com/tutorials/MySQLJava/article.html).

1. Install Eclipse on your computer
2. Download JDBC (Java DataBase Connectivity) from http://www.mysql.com/products/conector. Select "Platform Independent"
3. Unzip/unpack it. Open the new folder create by the unpacking. Make sure you see a jar file in the folder, with a name similar to mysql-connector-java-5.1.34-bin.jar
4. Create a new java project in Eclipse. Call it MySQLTest1 (or whatever name you want).
5. Drag & drop the JDBC connector jar file (mysql-connector-java-5.1.34-bin.jar) to your project in Eclipse. Select **copy file** and not "link file".
6. Create 2 java classes in the project (source code given below)

   Main.java, and
   MySQLAccess.java

7. Click on mysql-connector-java-5.1.34-bin.jar in the project to highlight it
8. Control-click or right-click on it to open the pop-up: pick **Build-Path**, and then **Configure Build-Path**
9. A new window opens up. In the right hand-side, click on the **Libraries** tab.
10. Click on **Add JARs**
11. In the JAR Selection window, double-click on MySQLTest1 (or whatever your **project** name is).
12. Select the **mysql-connector-java-5.1.34-bin.jar library**.
13. Click **Ok**.
14. The mysql connector is now a library available when compiling your java code.
15. Select your **Main** class in your project, in the Package Explorer window, and run it as an **Application**.
16. If everything goes well, you should see an output in your console, which is read directly from the database server in the lab.

```
User: lars
Website: http://www.vogella.com
Summary: Summary
Date: 2009-09-14
Comment: My first comment
User: lars
Website: http://www.vogella.com
Summary: Summary
Date: 2009-09-14
Comment: My first comment
User: Test
Website: TestWebpage
Summary: TestSummary
Date: 3910-01-11
Comment: TestComment
The columns in the table are:
Table: comments
Column 1 id
Column 2 MYUSER
Column 3 EMAIL
Column 4 WEBPAGE
Column 5 DATUM
Column 6 SUMMARY
Column 7 COMMENTS
```

Main.java

```
// Adapted from http://www.vogella.com/tutorials/MySQLJava/article.html
public class Main {
  public static void main(String[] args) throws Exception {
    MySQLAccess dao = new MySQLAccess();
    dao.readDataBase();
  }

}
```

MySQLAccess.java

```
// Adapted from http://www.vogella.com/tutorials/MySQLJava/article.html
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;

  final private String host = "xxxxxxxxxxxxxxxxxxxxxxxx";
  final private String user = "xxxxxxxx";
  final private String passwd = "xxxxxxxxxxx";

  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://" + host + "/feedback?"
              + "user=" + user + "&password=" + passwd );

      // 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) {

  }
}

}
```

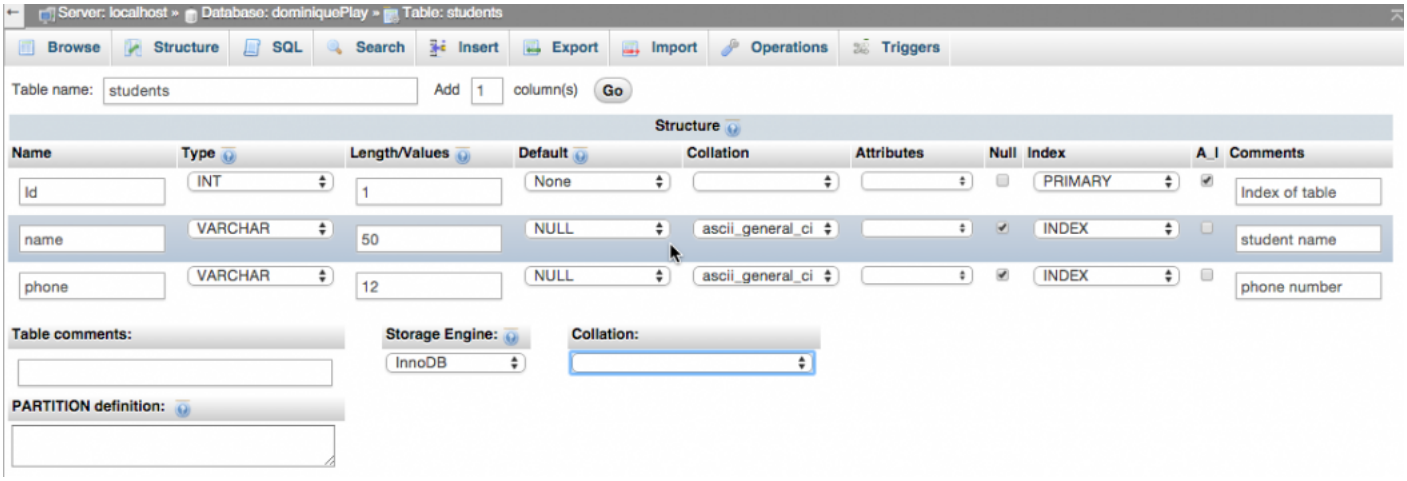
This section is only visible to computers located at **Smith College**

Setting up the Database

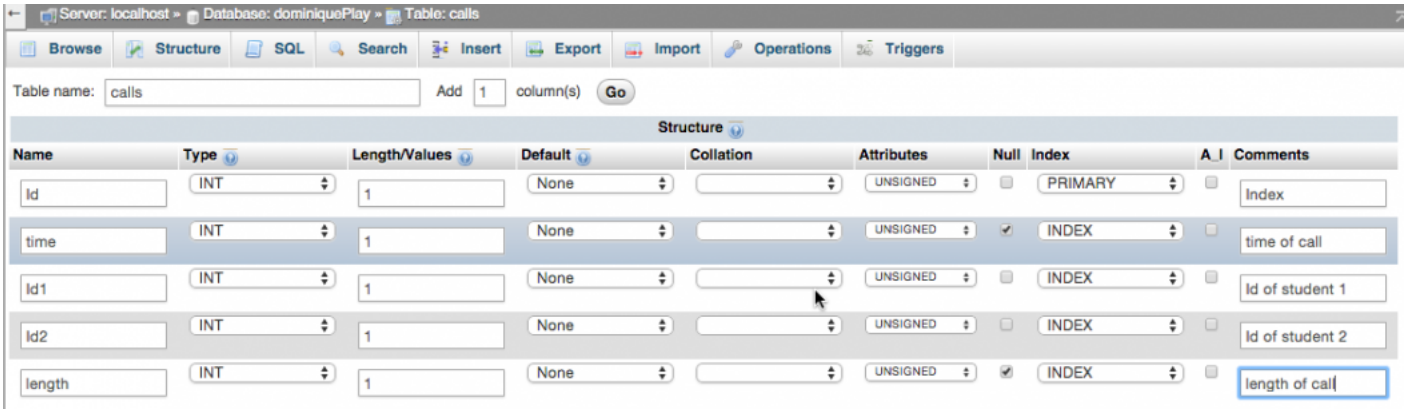
- Point your browser to a phpmyadmin installation that can access your MySQL server.

This section is only visible to computers located at **Smith College**

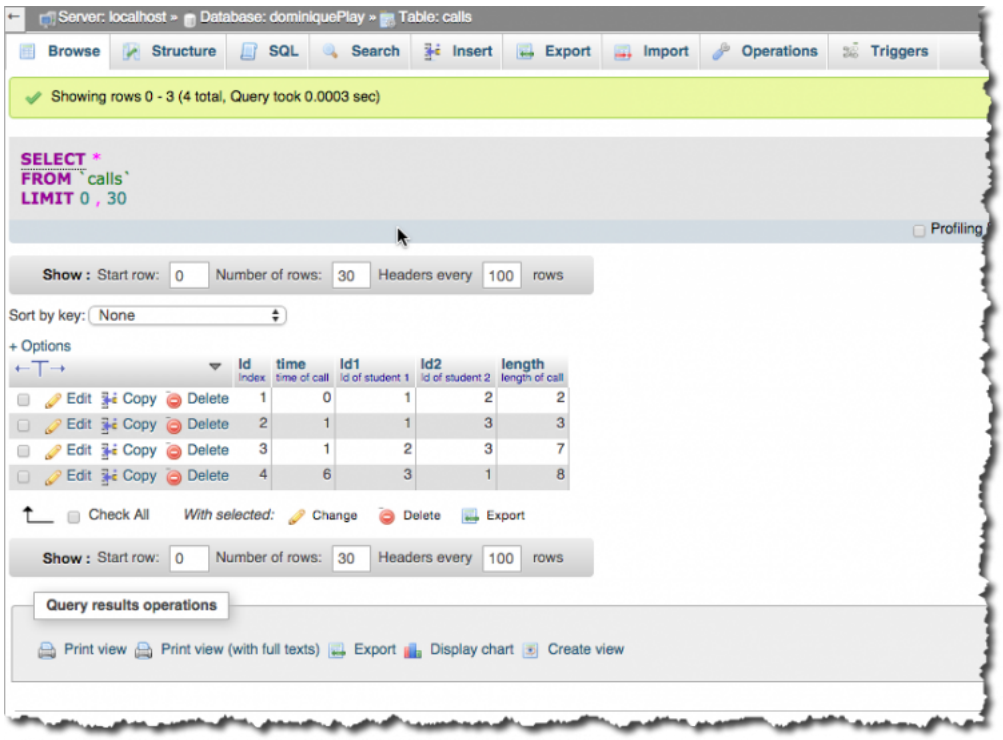
- Enter your MySQL account and password.
- In PhpMyAdmin, select **Create Table**, and call it **students**. Make it contain 3 columns.



- Create another table, called **calls** with 5 columns.(we are going to record phone calls).



- Populate the tables
  - Click on the **students** table in the left column, and then select the **Insert** tab.
  - Add 3 students: 1, yo, 1212, then 2, lu, 3434, and finally 3, do, 5656. You wil have to click **Insert** twice to enter all 3 students.
  - Click on **Browse**. Verify that you see the 3 students.
  - Click on the **calls** table in the left column, and then select the **Insert** tab.
  - Add 4 calls:
    - 1, 0, 1, 2, 2
    - 2, 1, 2, 3, 3
    - 3, 1, 2, 3, 7
    - 4, 6, 3, 1, 8
  - Click browse and verify that your table contains 4 calls, between different students, and with various call lengths.



Java Code to Access the New Database

- Create 2 new classes in your Java project:

AccessStudentsCalls.java

```
// Accesses a database with student and phone information

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 AccessStudentsCalls {

    private Connection connect = null;
    private Statement statement = null;
    private PreparedStatement preparedStatement = null;
    private ResultSet resultSet = null;

    final private String host = "xxxxxxxxxxxxxxxxxxxxx";
    final private String user = "xxxxxxxxxxxxxxxxxxxxx";
    final private String passwd = "xxxxxxxxxxxxxxxxxxxxx";
    final private String database = "xxxxxxxxxxxxxxxxxxxxx";

    public void connectToDB() 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://" + host + "/"
                    + database + "?" + "user=" + user + "&password=" + passwd);

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

    public void readStudents() throws Exception {
        try {
            statement = connect.createStatement();
            resultSet = statement
                    .executeQuery("select * from " + database + ".students");
            while (resultSet.next()) {
                int Id = resultSet.getInt("Id");
                String name = resultSet.getString("name");
                String phone = resultSet.getString("phone");

                System.out.println(String.format(
                        "Id: %d name: %5s  phone: %5s", Id, name, phone));
            }
        } catch (Exception e) {
            throw e;
        }
```

```
    }

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

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

            if (connect != null) {
                connect.close();
            }
        } catch (Exception e) {

        }
    }

}
```

MainStudentsCalls.java

```
public class MainStudentsCalls {
  public static void main(String[] args) throws Exception {
      AccessStudentsCalls db = new AccessStudentsCalls();
      db.connectToDB();
      db.readStudents();
      db.close();
  }

}
```

Testing, Challenges, Coding

- Run your program, and verify that you get the list of the students listed in your **students** table.

```
Id: 3 name:    do  phone:  5656
Id: 2 name:    lu  phone:  3434
Id: 1 name:    yo  phone:  1212
```

Challenge #1:

- Add a new method that will display the contents of the table **calls**.

Challenge #2:

- In the PhpMyAdmin window, click on the **SQL** tab.
- Enter this *query*:

```
SELECT * FROM  `calls`  WHERE length >5;
```

- Observe the result.
- Create a new method in your **AccessStudentsCalls** class that will display only the calls that are longer than *n*, where *n* is an int that will be passed to the method.

Challenge #3:

- Still in the **SQL** tab, in PhpMyAdmin, Try this query that contains a subquery (subqueries are documented on this page (http://dev.mysql.com/doc/refman/5.5/en/subqueries.html)):

```
SELECT * FROM calls WHERE Id1 = ( SELECT Id FROM students WHERE name =  "lu" );
```

- Add a new method that will display all the calls made **by** a student whose name (not Id) is passed to the method as a parameter.

- Add a new method that will display all the calls made **to** a student whose name (not Id) is passed to the medhod as a parameter.

Challenge #4:

- Add this new method to **AccessStudentsCalls.java**:

```
    public void addNewCall( int time, int Id1, int Id2, int length) throws Exception  {
        String query = "INSERT INTO `" + database + "`.`calls` (`Id`, `time`, `Id1`, `Id2`)
                     + "VALUES (NULL, ?, ?, ?, ? )";
        try {
            preparedStatement = connect.prepareStatement( query );
            preparedStatement.setInt(1, time );
            preparedStatement.setInt(2, Id1 );
            preparedStatement.setInt(3, Id2 );
            preparedStatement.setInt(4, length );
            preparedStatement.executeUpdate();

        } catch ( Exception e) {
            throw e;
        }
    }
```

- Call it from **MainStudentsCalls.java**, and make it add a new call, happening at Time 100, between Student 2 and Student 1, lasting 17 minutes.
- Run the program.
- Verify in **phpMyAdmin** that the new call has been recorded.
- Add a **for-loop** that will add 100 random calls between any of the 3 students, at different times, and lasting some different length of time. We don't care if the calls start at the same time or overlap.
- To generate a random int between **min** and **max**, use this *stackoverflow* solution (#363681):

```
public static int randInt(int min, int max) {

    // NOTE: Usually this should be a field rather than a method
    // variable so that it is not re-seeded every call.
    Random rand = new Random();

    // nextInt is normally exclusive of the top value,
    // so add 1 to make it inclusive
    int randomNum = rand.nextInt((max - min) + 1) + min;

    return randomNum;
}
```

References

- The *Bible* of all references: **dev.mysql.com** (http://dev.mysql.com/doc/refman/5.7/en/)
- Go through this excellent SQL tutorial by James Hoffman (http://cs.smith.edu/~thiebaut/tutorials/PhpMySql/SQLTutorial/sql1.html). Great way to become familiar with MySQL.
- I have several tutorials in this Wiki that might be of interest:
  - Lab in Php & MySQL (http://cs.smith.edu/~thiebaut/tutorials/PhpMySql/lab1.htm)
  - Lab in MySQL (http://cs.smith.edu/~thiebaut/tutorials/PhpMySql/lab2.htm)
  - Lab in MySQL on SELECT queries (http://cs.smith.edu/~thiebaut/tutorials/MySql/lab1.htm)
  - Lab in MySQL on JOIN operations (http://cs.smith.edu/~thiebaut/tutorials/MySql/lab2.htm)
  - A program for MPI that scans a directory for images (http://cs.smith.edu/dftwiki/index.php/Tutorial:_C_%2B_MySQL_%2B_MPI) and stores them in a database.

...
