Java Database Connection (JDBC)

Start mit MariaDB:

Man ladet MariaDB von

https://mariadb.org/

herunter. Ich habe die neueste Version und zwar das .msi-Format für die Windows-Installation benutzt. Die Installation war ziemlich geradlinig und hat dann sofort funktioniert. Ich habe bei der Installation als Datenformat UTF8 angekreuzt.

Mit dem "Command Prompt" Icon von MariaDB erhält man ein Window-Command Fenster, in welchem das bin-Directory von MariaDB bereits im Pfad inkludiert ist. Man kann dann damit direkt mysql aufrufen:

```
C:\Windows\system32>mysql -u root -p
```

Man muss dann noch das Passwort eingeben.

Nun erzeugt man die Datenbank:

```
CREATE DATABASE weiser1;
```

und verwendet sie anschließend:

```
USE weiser1;
```

Nun erzeugen wir eine Tabelle:

```
CREATE TABLE name1 (vorname VARCHAR(50),
   nachname VARCHAR(50), gebdat DATE);
```

und dann fügen wir mit einigen Kommandos Zeilen in die Tabelle ein:

```
INSERT name1 VALUES('Johann1', 'Weiser1', '1951-09-03');
```

Schließlich überprüfen wir, ob die Daten gespeichert sind:

```
SELECT * FROM name1;
```

Wir können auch eine sortierte Ausgabe erzeugen:

```
SELECT * FROM name1 ORDER BY gebdat;
```

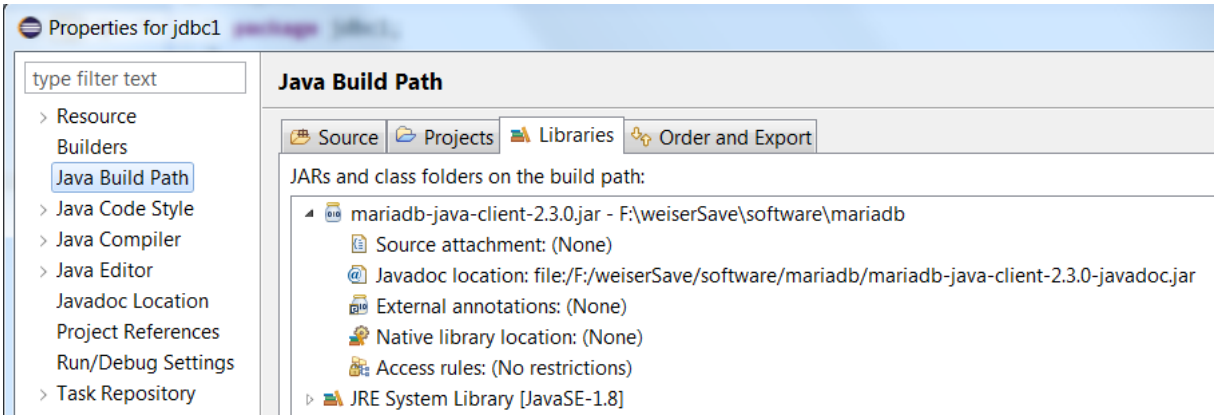
Mit JDBC auf MariaDB zugreifen:

Man benötigt zunächst den JDBC-Connector, es ist ein jar-File, ich ahbe die neueste 2.3-Serie heruntergeladen:

> https://downloads.mariadb.com/Connectors/java/connector-java-2.3.0/

und zwar eben das .jar-File und sinnvollerweise auch das dazugehörige javadoc.jar-File!

In Eclipse wird nun ein Projekt erstellt und der Connector als externes jar-File (inklusive der javadoc) im Java-Build-Path des Projektes eingehängt:



Hernach kann man ein einfaches Testprogramm beginnen. Man benötigt zunächst einmal die Klassen des packages java.sql:

```
import java.sql.*;
```

Ich habe eine eigene kleine Testmethode geschrieben und der Einfachheit halber den gesamten Code in ein try-catch Statement eingefügt, da verschiedene SQl-Exception abgefangen werden müssen:

```
public static void test1() {
   try {
      // hier steht der restliche Code!!
      ..............
   } catch (SQLException e) {
      e.printStackTrace();
   }
}
```

Man muss zunächst eine Verbindung mit der Datenbank herstellen:

```
Connection connection = DriverManager.getConnection(
   "jdbc:mariadb://localhost:3306/weiser1?" +
      user=root&password=Hallo1");
```

"mariadb" ist also so eine Art Schlüsselwort für die Datenbank, "localhost" die IP-Adresse des Datenbankservers, 3308 ist der zugehörige Server-Port, "weiser1" der Name der Datenbank und dann gibt es noch die Userid und das Passwort im Klartext. Ich habe den String hier in Dokument in zwei Teile unterteilt, da er in einer Zeile nicht Platz hat und ich einen gültigen Java-Code schreiben wollte!

Nun kann man ein SQL-Statement ausführen, hier ist es ein Query:

```
Statement s = connection.createStatement();
ResultSet rs = s.executeQuery("SELECT * from name1;");
```

Der Strichpunkt am Ende ist hier optional!

Das Ergebnis ist eine Art Liste von Zeilen, welche man in einer Schleife durchlaufen kann. Dabei werden jeweils die Daten der aktuellen Zeile ausgelesen:

```
while (rs.next()) {
   System.out.println(rs.getString("vorname") + " " +
      rs.getString("nachname") + ", geb. am " +
rs.getDate("gebdat"));
}
```

Am Ende sollte man noch die Datenbankresourcen freigeben. Dies ist zwar bei diesem Testprogramm egal, aber bei größeren Programmen ist dies durchaus wichtig:

```
rs.close();
s.close();
connection.close();
```