Datenbankzugriff mit Java (JDBC)

Von Christof Lieber – http://www.clieber.de/jdbc01.shtml

Als erstes geht es darum, den passenden Treiber zu finden. Hierbei handelt es sich um s.g. JDBC Treiber. Diese ermöglichen den Zugriff auf den Datenbanktyp für den sie geschrieben sind. Für erste Versuche ist hier der kostenlose MySQL-Server in der aktuellen Version zu empfehlen. Die aktuelle Version kann unter http://dev.mysql.com/downloads/mysql/ heruntergeladen werden.
Für MySQL stehen mehrere ebenfalls kostenloser Treiber zur Verfügung. Ich habe recht gute Erfahrungen mit dem MySQL Connector/J. Die aktuelle Version ist hier zu finden: http://dev.mysql.com/downloads/connector/j/.

Im Folgenden wird davon ausgegangen, dass die o.g Software in einer Windows-Umgebung genutzt wird. Links für andere Datenbanken sind unten aufgeführt.

Nachdem man die beiden Dateien heruntergeladen hat, starte man den Installer von MySQL. Eine sehr umfassende Dokumentation zu MySql inkl. Installationsanleitung findet man unter http://dev.mysql.com/get/Downloads/Manual/manual-a4.de.pdf/from/pick.
Als nächstes entpackt man die MySQL Connector/J Datei in ein beliebiges Verzeichnis. Hier befindet sich eine Datei mit dem Namen mysql-connector-java-*-bin.jar. Diese muss in den CLASSPATH abgelegt werden. Damit ist die Installation abgeschlossen.

Nun ist es Zeit unsere erste Datenbank und die erste Tabelle zu erstellen. Hierzu öffnen wir ein DOS-Fenster und wechseln in das MySQL Installationsverzeichnis.
Dann wechseln wir den Unterordner „bin". Während der Installation von MySQL wurde ein root-Passwort vergeben. Diese benötigen wir jetzt.
Nun wird die MySQL-Konsole mit dem Befehl

mysql -u root -p <<Passwort>>

gestartet. Nun sollte in der letzten Zeile etwa

mysql>

stehen.
Unsere erste Datenbank soll den Namen „Filme" haben. Sie wird mit folgendem Befehl angelegt:

create database filme;

Connect auf die Datenbank:

connect filme;

Jetzt legen wir unsere erste Tabelle an:

CREATE TABLE TEST_TABELLE(nr int, name char(255), sonstiges char(255));

Zur Erklärung: Der Befehl erzeugt eine Tabelle mit dem Namen TEST_TABELLE. Gleichzeitig werden folgende Spalten in der Tabelle angelegt: Die Spalte „nr" vom Type int sowie die Spalten „name" und „sonstiges" von Type char mit der Länge von 255 Zeichen angelegt. Falls keine SQL-Kenntnisse vorhanden sind lesen Sie bitte erst eines der Tutorials am Ende des Artikels. Im Folgenden werden SQL-Grundkenntnisse vorausgesetzt.
Auf der nächsten Seite werden wir uns nun endlich um die Implementierung in Java kümmern.
Nun zur Implementierung:

Folgendes Beispiel führt eine Beispielabfrage auf die eben angelegte Datenbank aus:

```
import java.sql.*;

public class MeineErsteAbfrage {

  Connection con = null;
  Statement stmt = null;
  ResultSet rs = null;

  public static void main(String args[]){
    try {
      Class.forName( "com.mysql.jdbc.Driver" ).getInstance();
    }
    catch ( ClassNotFoundException e ) {
      e.printStackTrace();
      System.exit(1);
    }
    try{
      con = DriverManager.getConnection("jdbc:mysql://localhost/filme",
              "root","<passwort>" );
      stmt = con.createStatement();
      rs = stmt.executeQuery(
              "SELECT nr,name,sonstiges FROM TEST_TABELLE");

      while ( rs.next() ){
        System.out.println("Nr:              "+rs.getString(1));
        System.out.println("Name:            "+rs.getString(2));
```

```
        System.out.println("Sonstiges:      "+rs.getString(3));
      }
      stmt.close();
      con.close();
    }
    catch ( SQLException e ){
     e.printStackTrace();
      return;
    }
  }
}
```

Erklärung:

Eine Abfrage mit JDBC läuft nach folgendem Muster ab:

1. Treiber bekannt machen
2. Verbindung zur Datenbank herstellen
3. Abfrage an Datenbank senden
4. Resultate verarbeiten
5. Verbindung abbauen

Der Treiber wird mit der Anweisung

```
Class.forName( "Name der Treiber-Klasse als String" ).getInstance();
```

bekannt gemacht.

In unserem Beispiel benutzen wir den Treiber, dessen Treiber-Klasse im Package *com.mysql.jdbc* liegt. Die Klasse selbst heißt *Driver*. Somit heißt bei uns die Anweisung

```
Class.forName( "com.mysql.jdbc.Driver" ).getInstance();
```

Als nächstes muss eine Verbindung aufgebaut werden. Dies geschieht mit der Anweisung

```
Connection con = DriverManager.getConnection(
        "jdbc:mysql://Server-Name/Datenbankname",
        "username","passwort" );
```

Für uns heißt das:

```
Connection con =
  DriverManager.getConnection("jdbc:mysql://localhost/filme",
        "user","Ihr_Passwort" );
```

Nun haben wir ein java.sql.Connection-Objekt. Mit diesem Objekt kann ein java.sql.Statement erzeugt werden, mit dem man dann SQL-Statements an die Datenbank senden kann. Die Erzeugung erfolgt wie folgt:


```
Statement stmt = con.createStatement();
```

Das Interface Statement hat einige Methoden, mit denen man Statements an die Datenbank senden kann, bzw. mehrere Statements zu einem Batch zusammenfassen kann. Hier eine kleine Auswahl:

- execute(String stmt)
- executeQuery(String stmt)
- executeUpdate(String stmt)
- addBatch(String stmt) u. executeBatch(String stmt)

Da wir in unserem ersten Beispiel ein SELECT-Statement benutzen wollen, verwenden wir die Methode *executeQuery()* und übergeben ihr das gewünschte Statement als String. Die Methode *executeQuery()* liefert ein so genanntes *ResultSet* zurück. Dieses enthält die zurückgelieferten Daten der Abfrage.

```
ResultSet rs = stmt.executeQuery("SELECT nr,name,sonstiges FROM
        TEST_TABELLE");
```

Nun muss eine Schleife nach dem Muster *while(rs.next())* gebaut werden. Die Methode *next()* liefert solange *true* zurück, bis es keine weitere Result-Zeile gibt und schieb den Zeilenzeiger, von vor der 1. Zeile anfangend, um eine Zeile nach unten. Innerhalb dieser Schleife kann dann mit den Methoden

- rs.getString(int Spaltennummer)
- rs.getInt(int Spaltennummer)
- rs.getDouble(int Spaltennummer)
- rs.getDate(int Spaltennummer)
- ...

über die Spaltennummer der Inhalt der einzelnen Zellen ausgeben. Zum Schluss wird mit folgendem Programm-Code noch aufgeräumt:

```
rs.close();
stmt.close();
con.close();
```

Das Tutorial wird ständig erweitert. Konstruktive Kritik ist sehr willkommen.

