# Oracle & Java HOW TO

Helge Janicke, Niels-Peter de Witt, Karsten Wolke

21. Januar 2002

## Inhaltsverzeichnis

# 1 Java-Anbindung an Oracle-DB

Zu allererst möchten wir hier auf eine Dokumentation hinweisen, die im Rahmen der Softwaretechnik- und Datenbankvorlesung bei Herrn Totzauer und Herrn Schiemann-Lillie entstanden ist.
In dieser Dokumentation wird beschrieben, wie eine Datenbank über die JDBC-Schnittstelle an eine Java-Applikation angebunden werden kann. Auch werden dort die wichtigsten und notwendigen Klassen und Interfaces genannt, und an kleinen Beispielen erläutert wie eine Anbindung zu realisieren ist. Wir empfehlen Ihnen sich dieses Dokument zu beschaffen um sich einen generellen Überblick zu schaffen.
Sie finden die Dokumentation unter:

http://www.furchur.de/de/dbsupport/Introduction.htm

# 2 Benötigte Programme und Daten

1. JDBC-Datenbank-Treiber für Ihre Oracle-Datenbank.
   Abhängig von dem Betriebssystem und der Oracle-Version, mit der Sie arbeiten bietet Oracle verschiedene Treiber an. Diese können Sie unter www.oracle.com erhalten. Leider müssen Sie sich dafür bei der Oracle-Community registrieren lassen. Diese Registrierung ist kostenlos. Die Community hat noch weitere Vorteile, Sie erhalten damit auch Zugriff auf viele Hilfen und Beispiele, die oft sehr nützlich sein können.
   Welche Version von Oracle Sie verwenden kann Ihnen am besten Ihr System-Administrator sagen. Wir haben für die im Fachbereich E&I verwendete Datenbank (Oracle Version 8.5) folgenden Treiber von Oracle heruntergeladen:

   - Oracle THIN Treiber.
     Ist ein Typ 4 Treiber, der gerade bei der Verwendung durch Applets o.ä. geeignet ist.

2. Informationen über die Datenbank.

   (a) Servernamen / Server - IP auf dem die Datenbank läuft.

   (b) Portnummer auf dem Sie eine Verbindung aufbauen können.

   (c) Den DSN (DataSourceName) der Datenbank.

   (d) Ihr Benutzername.

   (e) Ihr Passwort.

Nachdem Sie alle diese Informationen zusammengesucht haben kann es auch schon losgehen.

# 3 Einbinden der Klassen

Beiliegend zu dieser Dokumentation finden Sie ein Paket, daß die Oracle Treiber enthält. Um die Treiber verwenden zu können setzen Sie beim Übersetzen und Starten des Programms den `classpath` wie folgend:

Windows:
```
javac -classpath ".;./classes111.zip" Test.java
```
Linux:
```
javac -classpath ".:./classes111.zip" Test.java
```

Das Programm Test sollte sich nun übersetzen lassen. (Sie finden die Beschreibung des Beispiels im nachfolgendem Kapitel).

# 4 Aufbau einer Verbindung (Connection)

In Ihrer Applikation müssen Sie nun als erstes eine `Connection` aufbauen zu der Datenbank. Dies geschieht über folgende Zeilen:

```
String conURL   = "<Verbindungs-Informationen>";
String userName = "<Benutzer Name>";
String password = "<password>";
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection connection = java.sql.DriverManager.getConnection(conURL,
                        userName, password);
```

Die Verbindungs-Informationen sehen, bei der Verwendung des Oracle Thin Treibers, folgendermaßen aus:

```
conURL = "jdbc:oracle:thin:@<Server>:<Port>:<DSN-Name>";

          Verbindung wird über JDBC aufgebaut.
               Code des Treiberherstellers.
                      Treiberbezeichnung.
                            Name, bzw. IP des Servers.
                                      Port des Datenbankdienstes
                                      (Standardport: 1521)
                                             Name der DB
```

Hier ein Beispiel:

```
conURL = "jdbc:oracle:thin:@oracle.et-inf.fho-emden.de:1521:lab";
```

Diese Schritte dienten dem Einrichten und Öffnen einer Verbindung. Anschließend möchten wir Sie auf das Schließen einer Verbindung hinweisen. Dies ist wichtig, damit evtl. geänderte Daten auch in die Datenbank übernommen werden (*commit*).
Das Schließen erfolgt über die Verbindung `connection`

```
connection.close();
```

und sollte spätestens am Ende Ihres Programmes erfolgen.

**HINWEIS:**
Beim Öffnen und Schließen, sowie allen anderen Aktionen, die die Connection benutzen, können SQLExceptions auftreten und müssen abgefangen werden.

## 4.1 Verbindungsaufbau an einem Beispiel

Wenn Sie die folgende kleine Datei erstellen, können Sie testen, ob Ihr Treiber korrekt ist, sowie ob Sie mit Ihrem Benutzernamen und Password auf die angebene Datenbank zugreifen dürfen.

```
// ----------------- Beispiel -------------------------------------------
import java.sql.*;
import java.util.*;

class Test {

  // Bitte Ihren Namen statt <Ihr Benutzername> angeben
  String userName = "<Ihr Benutzername>";

  // Bitte Ihr Password statt <Ihr Password> angeben
  String password = "<Ihr Password>";

  // Der String zur Beschreibung der Verbindung.
  String conURL = "jdbc:oracle:thin:@oracle.et-inf.fho-emden.de:1521:lab";

  // Die Verbindung zu DB
  Connection connection = null;

  public Test() {
    try {
      // Öffnen der Verbindung
      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
      connection =
         java.sql.DriverManager.getConnection(conURL, userName, password);

      System.out.println("Verbindung aufgebaut");

      // LESEN / SCHREIBEN SIE HIER VON / IN DIE DB
      System.out.println("später greifen Sie hier auf die DB zu");

      // Schließen der Verbindung
      connection.close();
      System.out.println("Verbindung abgebaut");

    } catch (Exception exc) {
           System.err.println("Es ist ein Fehler aufgetreten:\n" +
                       exc.getMessage());
      exc.printStackTrace();
      try {
        // Wenn ein Fehler auftritt, Fehler ausgeben und versuchen
        // die Datenbank-Verbindung zu schließen.

        connection.close();
        System.err.println("Verbindung abgebaut");
      } catch (SQLException sqlexc) {
                System.err.println("Verbindung konnte nicht geschlossen werden.");
            } catch (NullPointerException nulexc) {
                System.err.println("Es wurde keine Verbindung geoeffnet.");
            }
    }
    }

  public static void main(String[] args) {
    Test t = new Test();
  }
}
// ----------------- Beispiel Ende ---------------------------------------
```

[Test.java]

Wenn Sie dieses Programm fehlerfrei übersetzen konnten, und die Meldung:

```
Verbindung aufgebaut
später greifen Sie hier auf die DB zu
Verbindung abgebaut
```

sehen gibt es keine Probleme mehr (jedenfalls nicht mit der Datenbank-Verbindung).

# 5 Versenden von SQL Anweisungen

Nachdem Sie die Verbindung zu Ihrer Datenbank aufgebaut haben können Sie auf die dort vorhandenen Daten zugreifen. Dazu werden sogenannte `Statement`s erzeugt.

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

Diese Zeile erzeugt sozusagen ein leeres Statement. Für eine genaue Beschreibung des interfaces schlagen Sie bitte in der Java API Dokumentation nach. Eine SQL Anweisung können Sie dann wie folgend erstellen:

```
String sqlInsert    = "INSERT INTO Kunde VALUES ('TestName',100)";
```

Jede Anfrage an die Datenbank, die Daten verändert (INSERT INTO), erzeugt (CREATE, GRANT ...) oder löscht (DELETE) sind Update-Anweisungen.

Update-Anweisungen werden mit folgender Methode ausgeführt:

```
statement.executeUpdate(sqlInsert);
```

Oft will man nur lesend (`"SELECT * FROM Kunde"`) auf die Datenbank zugreifen und auch ein Ergebnis erhalten. Für solche Anfragen (Queries) verwenden Sie die Methode:

```
String sqlQuery = "SELECT * FROM Kunde";
ResultSet result = null;
result = statement.executeQuery(sqlQuery);
```

Die Zeilen für das Abfangen einer SQLException haben wir der Übersichtlichkeithalber weggelassen. Grundsätzlich läßt sich aber sagen, daß fast jede Methode des Paketes java.sql eine SQLException erzeugen kann.
Die Auswertung des Ergebnisses der SQL Abfrage entnehmen Sie bitte auch dem Eingangs genannten Skript oder der Java API Dokumentation. An dieser Stelle nur eine Methode zur Anzeige eines allgemeinen `ResultSet`s:

```
public void showResult(ResultSet result) {
  int counter = 0;
  StringBuffer text = new StringBuffer("RESULT\n");
  try {
    // While there is a set of data in the ResultSet add the
    // data to text
    while (result.next()) {
      counter++;
      text.append("****************************************\n");
      text.append("* SET No. "+counter+"\n");
      text.append("****************************************\n");
      // putting all columns of the ResutSet data to text
      for (int i=1; i<=getColumnCount(result); i++) {
        try {
          // result.getObject(<column-index>) returns the Object
          // from the given column in the actual used row of data.
          Object col_Item = result.getObject(i);
          if (col_Item == null) {
            text.append("\tcontains = >null<\n");
          } else {
            text.append("\tcontains = " +
                        result.getObject(i).toString()+"\n");
          }
        } catch (Exception exc) {
          exc.printStackTrace();
        }
      }
    }
    System.out.println(text.toString());
  } catch (SQLException sqlexc) {
    sqlexc.printStackTrace();
  }
}
```