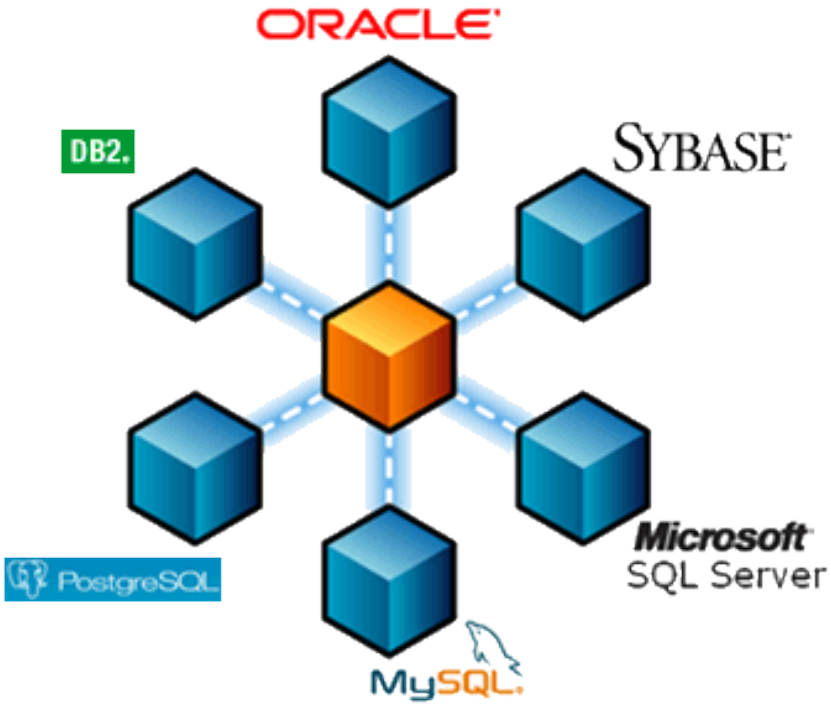
RELATIONONALE DATENBANKEN MIT JDBC



Christoph Süsens

07.05.2012

Inhalt

Abbildungsverzeichnis 3
Java Database Connection – JDBC 4
Herstellen einer Verbindung JDBC - DBMS 4
Treiber Installation Beispiel: Sybase SQL Anywhere 5
Netzwerkserver starten 5
Programmieren der Connection JAVA Klasse 7
Code Sample JAVA 8
Musikdatenbank 10
Eclipse Data Tools Platform – Data Source Explorer 11
Literaturverzeichnis 13

Abbildungsverzeichnis

Abbildung 1 - Beispiel JDBC Treiber 4.0 4
Abbildung 2 - Setzen des CLASSPATH für JDBC Treiber 5
Abbildung 3 - Startoptionen SQL Anywhere Netzwerkserver 6
Abbildung 4 -Netzwerkserver mit Portangabe 6
Abbildung 5 - ER-Diagramm Musikdatenbank 10
Abbildung 6 - Database Connection herstellen 11
Abbildung 7 - New Driver Difinition 11
Abbildung 8 - Connection Konfiguration 12

Java Database Connection – JDBC

Java Database Connection, kurz JDBC ist eine Datenbankschnittstelle auf der Java-Plattform. Diese bietet eine Zugriffmöglichkeit auf relationale Datenbanken verschiedener Datenbanksysteme wie z. B. Sybase SQL Anywhere.

Voraussetzung dafür ist ein JDBC Treiber, dieser stellt eine direkte Verbindung zum Datenbankmanagementsystem (DBMS) her. So können SQL-Anweisungen direkt aus einem Java-Programm an das DBMS gesendet und verarbeitet werden.[1]

Herstellen einer Verbindung JDBC - DBMS

Damit eine Verbindung zum DBMS hergestellt werden kann, muss ein JDBC Treiber installiert werden, in dieser Dokumentation wird das DBMS Sybase SQL Anywhere 12 und der JDBC Treiber Typ 4 benutzt. Typ 4 ist ein Nativer Protokoll Driver, damit wird eine direkte Verbindung ohne Middleware auf das DBMS hergestellt.[2]

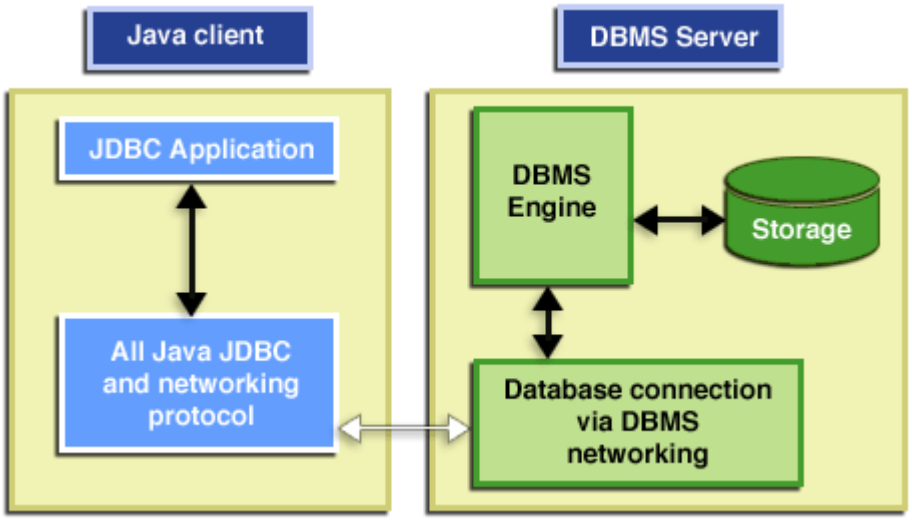


Abbildung 1 - Beispiel JDBC Treiber 4.0[3]

[1] http://de.wikipedia.org/wiki/Java_Database_Connectivity 04.04.12 21:20 Uhr
[2] http://de.wikipedia.org/wiki/Java_Database_Connectivity 04.04.12 21:39 Uhr
[3] http://fpetito.blogspot.de/ 04.04.12 21:43 Uhr

Treiber Installation Beispiel: Sybase SQL Anywhere

Der JDBC Treiber der benötigt wird für Sybase SQL Anywhere kann über die Internetseite http://www.sybase.com/products/allproductsa-z/softwaredeveloperkit/jconnect bezogen werden.

Nach einer kurzen Registrierung und dem Download der benötigten JConnect Datei, muss ein CLASSPATH zum Installationsverzeichnis der JConnect Datei gesetzt werden.

Um den CLASSPATH zu setzen muss unter Systemumgebung „System“ geöffnet werden, auf der linken Seite den Menüpunkt „Erweiterte Systemeinstellungen“ aufrufen. Dort erscheint unten rechts ein neuer Menüpunkt „Umgebungsvariablen“, diesen öffnen. Bei Systemvariablen den Menüpunkt „NEU“ wählen. Name der Variablen lautet CLASSPATH; Wert der Variable besteht aus den Pfad zum JDBC Treiber Typ 4.

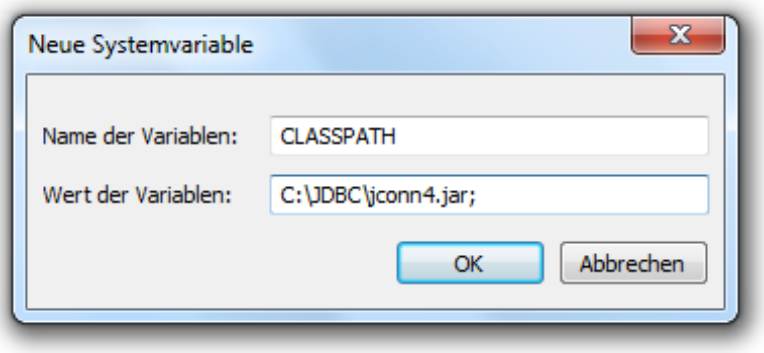


Abbildung 2 - Setzen des CLASSPATH für JDBC Treiber

Alternativ kann über Eingabeaufforderung (cmd.exe) mittels „set CLASSPATH=Pfad\jconn4.jar;“ der CLASSPATH gesetzt werden. Mittels echo %CLASSPATH% kann geprüft werden, ob die Eintragung erfolgreich war.[4]

Netzwerkserver starten

Nachdem der JDBC Treiber erfolgreich installiert wurde, muss der Netzwerkserver von SQL Anywhere gestartet werden, der Aufruf erfolgt über Start => Sybase SQL Anywhere 12 => SQL Anywhere => Netzwerkserver. In den Startoptionen für den Server (Abbildung 3) muss die zu landende Datenbank ausgewählt und ein Servername angegeben werden. Mit OK wird der Netzwerkserver gestartet, dieser kann nun über den Port 2638 (Abbildung 4) später mittels des JAVA Programms angesprochen werden und ermöglicht den direkten Zugriff auf die Datenbank.

[4] http://www.sumedh.info/articles/java-jdbc-sybase.php 04.04.12 22:09

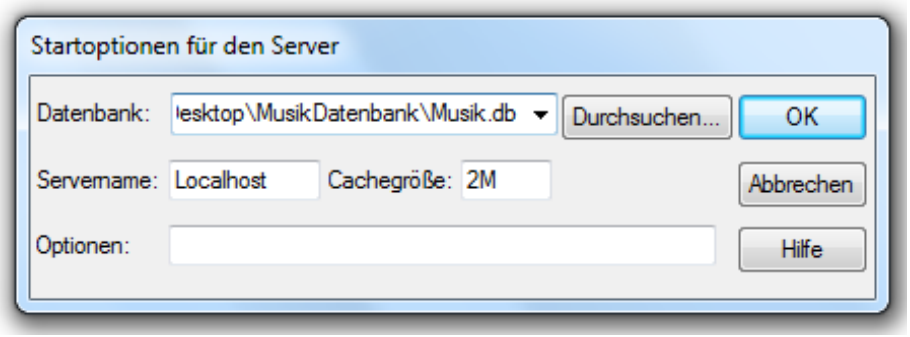


Abbildung 3 - Startoptionen SQL Anywhere Netzwerkserver

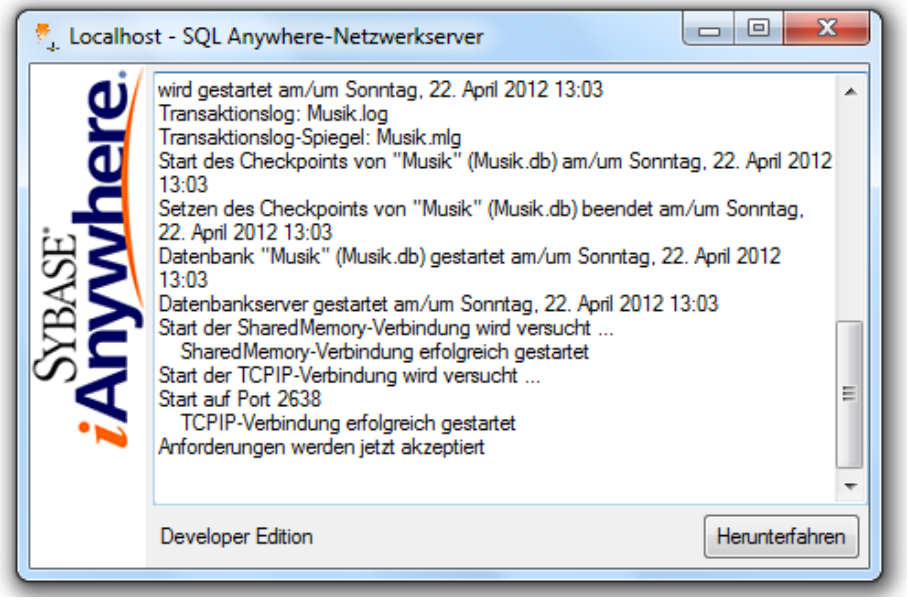


Abbildung 4 -Netzwerkserver mit Portangabe

Achtung:

Sollte der Netzwerkserver nicht gestartet worden sein, so erscheint später beim Ausführen der Java Klasse die Fehlermeldung „`JZ006: Caught IOException: java.net.ConnectException: Connection refused: connect`“.

Programmieren der Connection JAVA Klasse

Damit das Java-Programm mit der JDBC Schnittstelle kommunizieren kann, muss über „**Import java.sql.***“ das benötigte Package in die Klasse importiert werden. Darin enthalten ist der sog. DriverManger, welcher die JDBC Treiber verwaltet. Der DriverManger ist für das Laden der Treiber verantwortlich, um später eine Verbindung mit der Datenbank zu ermöglichen. Das Laden erfolgt über den Ausdruck Class.forName(„Treiber-Name“).

Für Sybase SQL Anywhere 12 wird folgender Ausdruck benötigt:

```
Class.forName("com.sybase.jdbc4.jdbc.SybDriver");
```

Nun kann eine Verbindung kreiert werden. Dies erfolgt über ein Objekt vom Typ Connection. Diese Aufgabe delegiert man an den DriverManger, indem man ihm über die Operation getConnection() die Adresse der Datenbank, in Form einer URL übergibt, gefolgt von Benutzername und Passwort falls notwendig.

```
Connection conn = DriverManger.getConnection(DB_URL, name, passwd);
```

Beispiel Sybase SQL Anywhere 12:

```
Connection conn = DriverManager.getConnection
("jdbc:sybase:Tds:localhost:2638?ServiceName=musik","dba","sql");
```

Die localhost Variable kann auch durch eine IP-Adresse zum Zielserver ersetzt werden, danach folgt der Port des Netzwerkservers. Wichtig dabei ist, dass später beim Ausführen des Java-Programms der Netzwerkserver auch online ist.

Mit diesen beiden Einstellungen sollte nun eine erfolgreiche Verbindung zur Datenbank zustande gekommen sein. Damit nun SQL Anweisungen an das DBMS abgesetzt werden können, muss ein Objekt vom Typ Statement generiert werden. Dies erledigt die Operation createStatement der Klasse Connection wie folgt:

Statement stmt = conn.createStatement( );

Mittels **executeQuery()** können Anfragen- und über **executeUpdate()** können Änderungsoperationen durchgeführt werden.

Ein Beispiel für eine executeQuery() könnte folgendermaßen aussehen:

```
ResultSet rs = stmt.executeQuery("select KundenNr, Firma from Kunde");

while(rs.next())
{
      System.out.println(rs.getString("KundenNr") + "" + rs.getString("Firma"));
}
```

ResultSet stellt eine Iterator-Schnittstelle für den Zugriff auf die Menge von Ergebnistupeln der SQL-Abfrage bereit. Die Operation .next() ruft das nächste Tupel auf. Mittels der while Schleife wird so jeder Tupel iteriert. Über .getString() kann auf das aktuelle Attribut des Ergebnistupel zugegriffen werden.

Statt eines .getString("String") kann auch einfach die Position des Attributes eingetragen werden z. B. .getString(1). Sollen die ausgelesen Daten weiter verarbeitet werden, so können die Ergebnisse direkt in entsprechende Java-Objekte bzw. Variablen kopiert werden.

Nach Abschluss der Verarbeitung sollte das Statement und die Verbindung zur Datenbank wieder geschlossen werden. Dazu wird die close() Operation verwendet.[5]

Beispiel:

```
if (rs != null) rs.close();
if (stmt != null) stmt.close();
if (conn != null) conn.close();
```

Code Sample JAVA

Nachfolgend das gesamte Code Beispiel für eine Anwendung von JDBC mit JAVA und relationalen Datenbanken. Die Einbindung von Try und Catch Anweisungen ist zwingend Vorgeschrieben, da die Exceptions wenig aussagekräftig sind, sollten diese angepasst werden.

```
import java.sql.*;

public class Start {

	/**
	 * example for jdbc connection
	 */
	public static void main(String[] args) {

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


		/**registered driver from sybase
		 * so no import com.sybase.jdbc.* is necessary
		 * attention classpath to jdbc must be set
		 */
		try{

			DriverManager.registerDriver((Driver)
			Class.forName("com.sybase.jdbc4.jdbc.SybDriver").newInstance())
			;
		}
		catch(Exception e){
			System.out.println(e.getMessage());
		}
```

[5] Alfons Kemper, André Eickler - Datenbanksysteme Eine Einführung 8. Auflage – Seite 145 Kapitel 4.21.1

```
		/**
		 * connect to database
		 */
		try{

			con = DriverManager.getConnection(
				"jdbc:sybase:Tds:localhost:2638?ServiceName
					=vorlesung","dba","sql");

		/**
		 * create statement to send to database
		 * and execute a sql query
		 */

			stmt = con.createStatement();

			rs = stmt.executeQuery("select KundenNr, Firma from Kunde");
			while(rs.next()){
				System.out.println(rs.getString("KundenNr") + "" +
				rs.getString("Firma"));
			}
		}
		catch(Exception e){
			System.out.println(e.getMessage());
		}

		/**
		 * set resources free
		 */
		try{

			if (rs != null) rs.close();
			if (stmt != null) stmt.close();
			if (con != null) con.close();

		}catch (Exception e){
			System.out.println(e.getMessage());
		}
	}
}
```

Musikdatenbank

Am Beispiel einer Musikdatenbank soll weitestgehend verdeutlich werden wie genau Java mit JDBC und SQL Abfragen miteinander kombiniert werden kann. Dazu muss zuerst in Sybase SQL Anywhere 12 eine neue Datenbank, nach dem unten in Abbildung 5 dargestellten Datenbankschema erstellt werden.

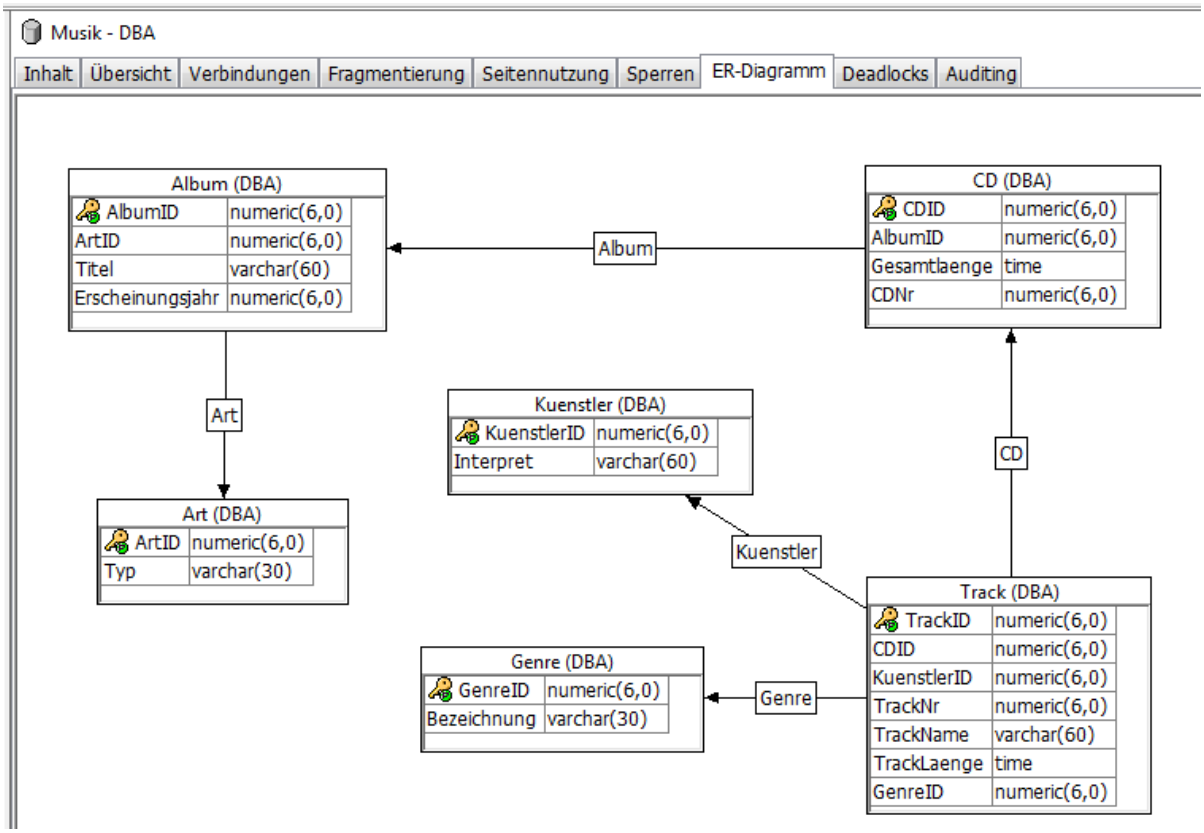


Abbildung 5 - ER-Diagramm Musikdatenbank

Eine neue Datenbank wird über Start -> SQL Anywhere 12 -> Administrationstools -> Sybase Central -> Extras -> Datenbank erstellen erzeugt, als Datenbankname wird “Musik“ vergeben. Um die Erzeugung der benötigten Tabellen und Beispieleinträge zu erleichtern, sollte das mitgelieferte SQL Skript verwendet werden. Das Einlesen erfolgt über Extras -> Interactive SQL öffnen, das bereitgestellte SQL-Skript ausführen.

Eclipse Data Tools Platform – Data Source Explorer

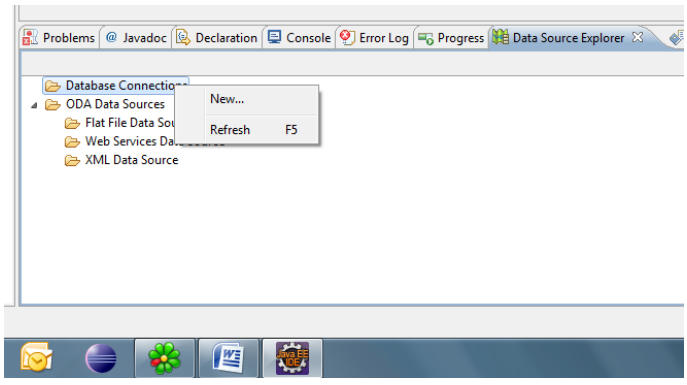
Zum Schluss soll noch ein Plugin mit dem Namen „Data Source Explorer“ vorgestellt werden, dieses Tool ermöglicht es direkt in Eclipse auf die Datenbank zuzugreifen ohne das benötigte DBMS aufrufen zu müssen. Damit dies funktioniert, muss der Reiter „Help“ geöffnet und dort der Menüpunkt „Install New Software…“ aufgerufen werden.

Ein Klick auf „Find more software by working with the “Available Software Sites” preferences” liefert eine Auswahlliste. Nach Eingabe des Suchbegriffes “DTP” erscheint dort der benötigte Link, diesen Aktivieren und mit OK bestätigen.

Zurück im vorherigen Menüfenster, unter „Work with“ muss nun der Eintrag „DTP Update Site - http://download.eclipse.org/datatools/updates“ ausgewählt werden. Im freien Textfeld mit der Bezeichnung „type filter text“ muss das Wort „data“ eingetragen werden. Es erscheint wieder eine Auswahlliste, bis zum Eintrag „Enabling Features for DTP 1.6.0“ scrollen und den Eintrag „Data Tools Platform Enablement for Sybase“ auswählen. Mit Bestätigung auf „Next“, wird der Data Source Explorer installiert.

Data Source Explorer aktivieren über den Reiter „Windows“ -> „Show View“ -> „Other“ -> „Data Management”. Neben dem Data Source Explorer steht noch ein weiteres nützliches Tool zur Verfügung „SQL Results“, dieses ermöglicht das Ausführen von SQL Anweisungen, wird jedoch nicht näher beschrieben.

Damit der Zugriff auf die Datenbank funktioniert, muss im Data Source Explorer erst eine Verbindung zur Datenbank hergestellt werden. Die Eintragungen sind die Selben wie für JDBC. Im Data Source Explorer den Ordner „Database Connection“ mit der Rechtenmaustaste auswählen und über New.. eine neue Verbindung auswählen, siehe Abbildung 6 – Database Connection herstellen.



Es erscheint ein neues Menü, dort das Connection Profile für Sybase ASA auswählen und bei Name, den Namen der Datenbank eintragen und Next drücken. Es muss der JDBC Treiber ausgewählt, bzw. neu Eingetragen werden; Abbildung 7 – New Driver Definition.

Abbildung 6 - Database Connection herstellen

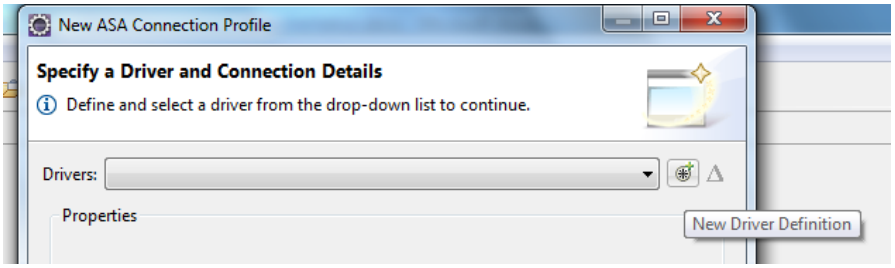


Abbildung 7 - New Driver Difinition

Im nächsten Menüfenster die geeignete Datenbank auswählen, im Reiter „JAR List“ den JConnect 4 Treiber hinterlegen und im Reiter „Properties“, die JDBC Connection Informationen eingeben. Das Ganze bestätigen und wenn gewollt, das Passwort speichern.

Mittels Test Connection kann geprüft werden, ob die Eintragungen korrekt sind, dabei sollte aber beachtet werden, dass der Netzwerkserver von Sybase auch online ist; Abbildung 8 – Connection Konfiguration. Mit Finish wird die Konfiguration abgeschlossen und im Ordner Database Connection sollte nun die Datenbank hinterlegt sein.[6]

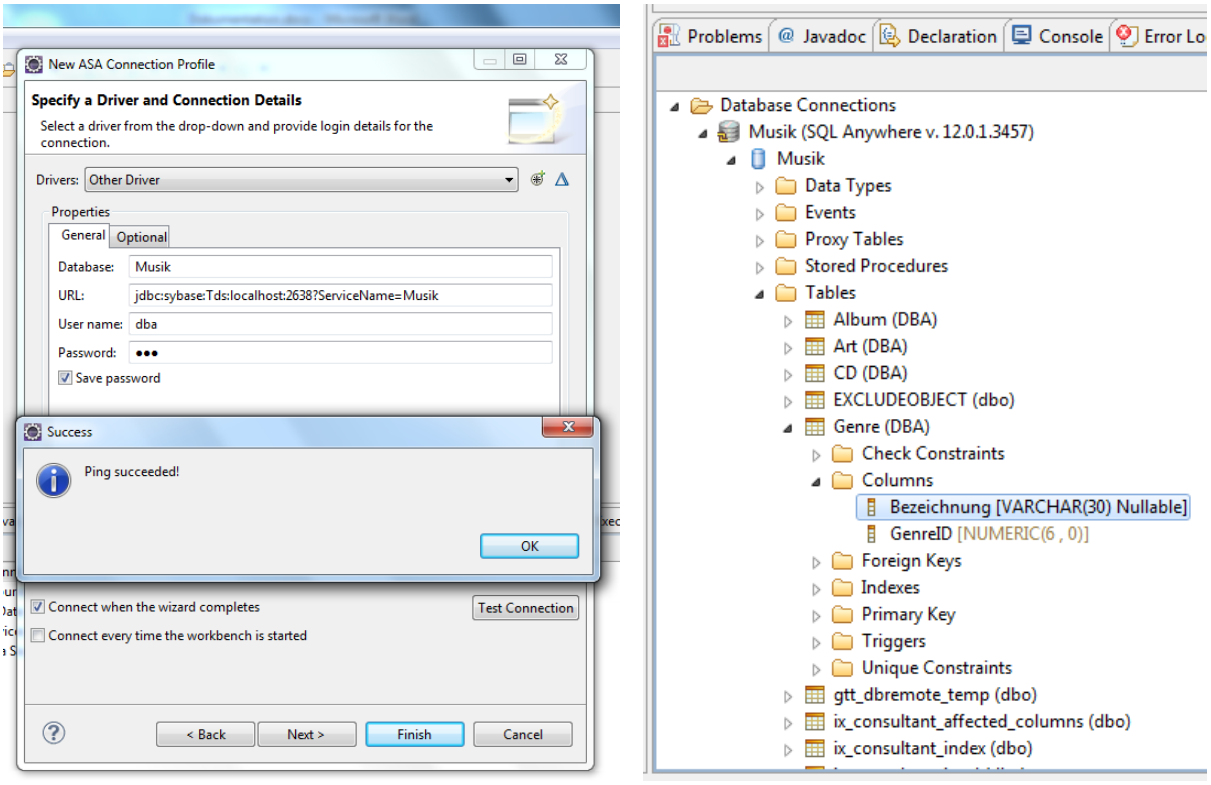


Abbildung 8 - Connection Konfiguration

[6] http://openbook.galileocomputing.de/javainsel/javainsel_17_003.html 17.04.2012 18:38 Uhr

Literaturverzeichnis

- http://de.wikipedia.org/wiki/Java_Database_Connectivity 04.04.12 21:20 Uhr
- http://de.wikipedia.org/wiki/Java_Database_Connectivity 04.04.12 21:39 Uhr
- http://fpetito.blogspot.de/ 04.04.12 21:43 Uhr
- http://www.sumedh.info/articles/java-jdbc-sybase.php 04.04.12 22:09 Uhr
- Alfons Kemper, André Eickler - Datenbanksysteme Eine Einführung 8. Auflage - ISBN 978-486-59834-6
- http://www2.cs.uni-paderborn.de/cs/ag-boettcher/lehre/WS0304/dbis1/download/dbis1k4-jdbc/dbis1k4-jdbc.pdf 15.04.2012 13:59 Uhr
- http://openbook.galileocomputing.de/javainsel/javainsel_17_003.html 17.04.12 18:38 Uhr
- http://www.juergen-bayer.net/artikel/Java/JDBC/JDBC-Grundlagen.aspx 03.05.12 19:56 Uhr