Grafische Nutzerschnittstellen

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt

Java und Datenbanken

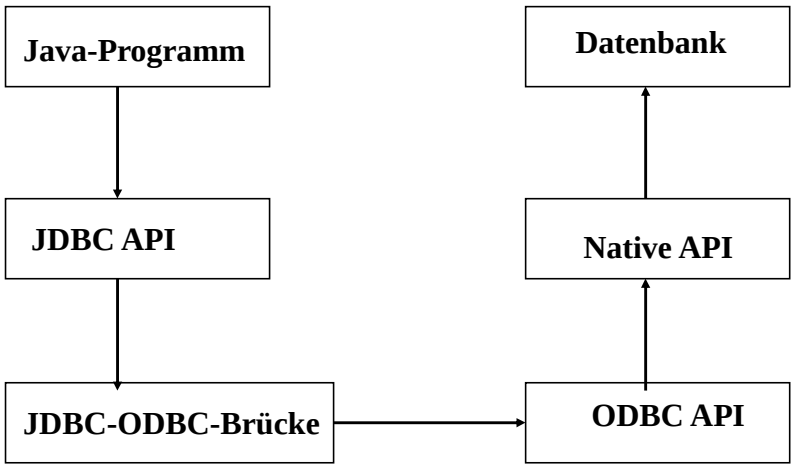
Methoden

- **JDBC mit ODBC-Brücke (Typ 1)**
- **JDBC zu nativer API (Typ 2)**
- **JDBC zu Treiber-Server (Typ 3)**
- **JDBCpur (Typ 4)**

Quelle: Java 5, Markt & Technik

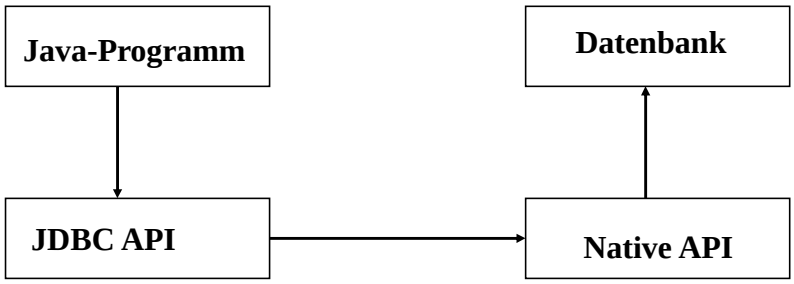
Open Database Connectivity

JDBC mit ODBC-Brücke, bis zu 20x langsamer



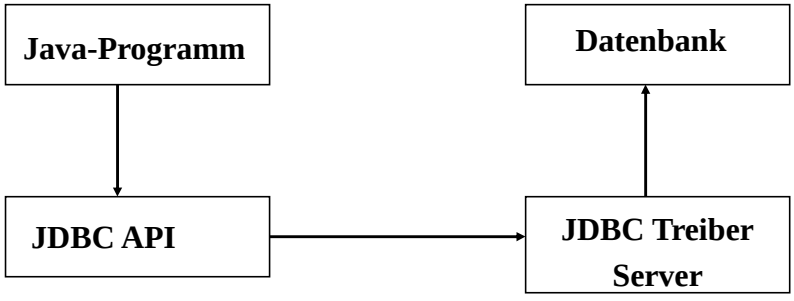
Spezielle Treiber sind erforderlich, veraltet

JDBC zu nativer API (Typ 2)



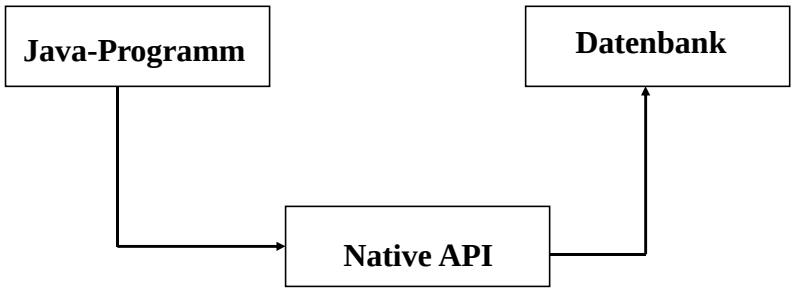
Erläuterung mit Beispiel
Spezielle Treiber sind erforderlich

JDBC zu Treiber-Server (Typ 3)



Zentrale Softwareschicht beim Server
Gut für Applets

JDBCpur (Typ 4)



Nur Java-Source wird benötigt.
Gut für Applets

Aufbau JDBC

- **Import**
  - import java.sql.*;
- **Anmeldedaten**
  - sDbDrv = "org.firebirdsql.jdbc.FBDriver"
  - sDbUrl="jdbc:firebirdsql:localhost/3050:D:\\dbs\emp.fdb";
  - sTable="employee";
  - sUsr="sysdba";
  - sPwd="masterkey";
  - Connection cn = null;
  - Statement st = null;
  - ResultSet rs = null;

Aufbau JDBC

- **Aufbau der Verbindung**
  - Class.forName(sDbDrv);
  - cn = DriverManager.getConnection(sDbUrl, sUsr, sPwd);
  - st = cn.createStatement();
  - rs = st.executeQuery("select * from " + sTable+" where salary > 70000.0");

Die Klasse “Class” stellt Informationen über eine Klasse zur Verfügung, also Metadaten. Des Weiteren erlaubt sie aber, direkt neue Instanzen zu erzeugen, also OHNE new. Diese Technik wird in Plugin´s benötigt

- **Konstruktor:**
- Class I = Class.forName("java.lang.Integer");
- Class<Integer> classObjekt = einObjekt.getClass();
- Class<?> classObjekt = Class.forName("java.lang.Integer");

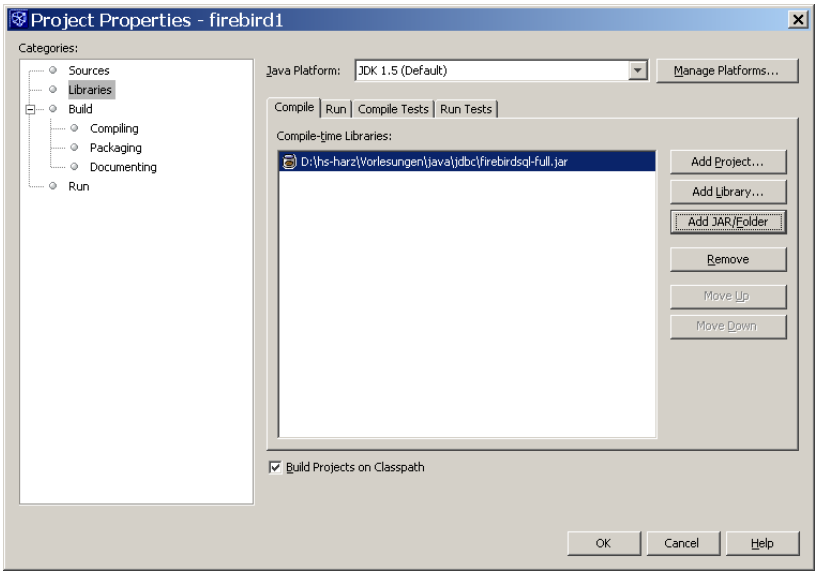
- **Beispiel:**
- Class myclass = Class.forName("javax.swing.JFrame");
- JFrame f = (JFrame) myclass.newInstance();
- f.setTitle("Class.forName");
- f.setSize(100,200);
- f.setLocation(400,100);
- f.setVisible(true);

Aufbau JDBC

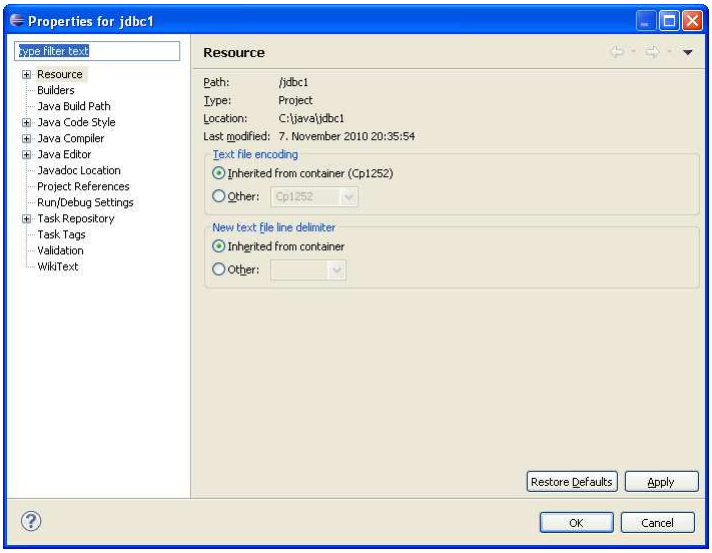
- **Auswertung der SQL-Anweisung**

```
ResultSetMetaData rsmd = rs.getMetaData();
int i, n;
n = rsmd.getColumnCount();   // Anzahl der Spalten  1 bis n
for( i=1; i<=n; i++ ) {
    System.out.print(  rsmd.getColumnName( i )    );
}

System.out.println( " " );  // Ausgabe der Tupel, Anzahl der Zeilen unbekannt
while(   rs.next()    ) {   //
    for( i=1; i<=n; i++ )       // Tupel ausgeben
    System.out.print(   rs.getString( i )    );
}
```

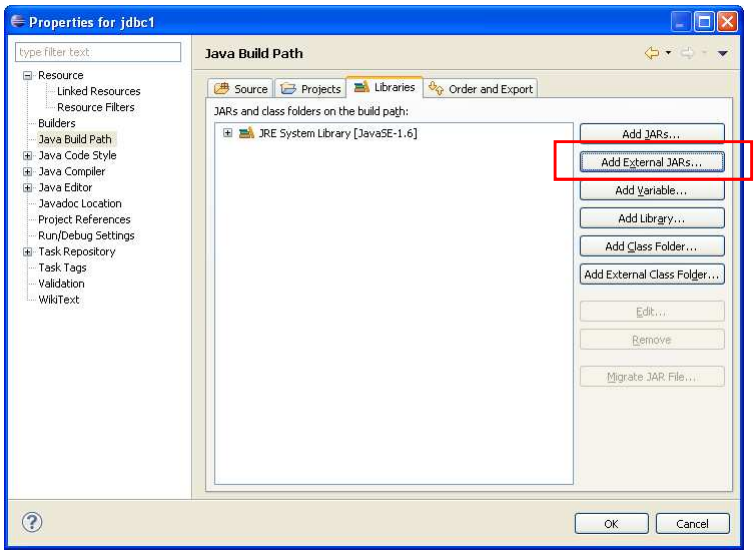

DBS-Treiber in Netbeans



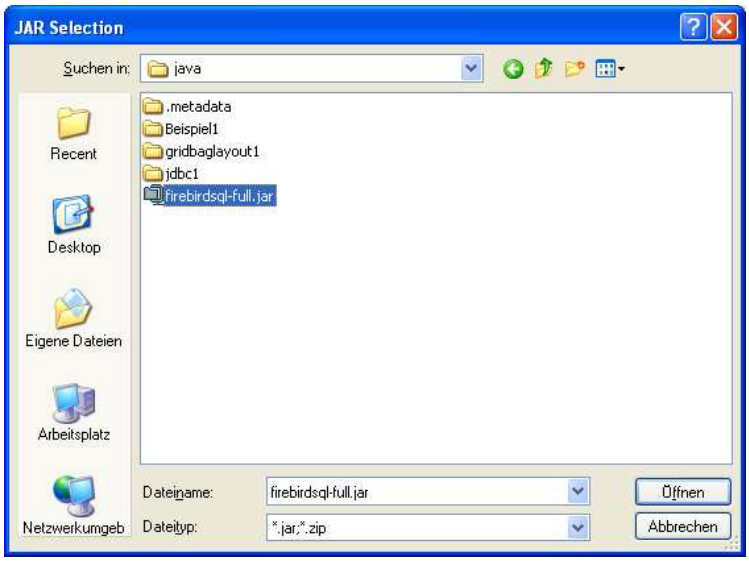
DBS-Treiber in Eclipse



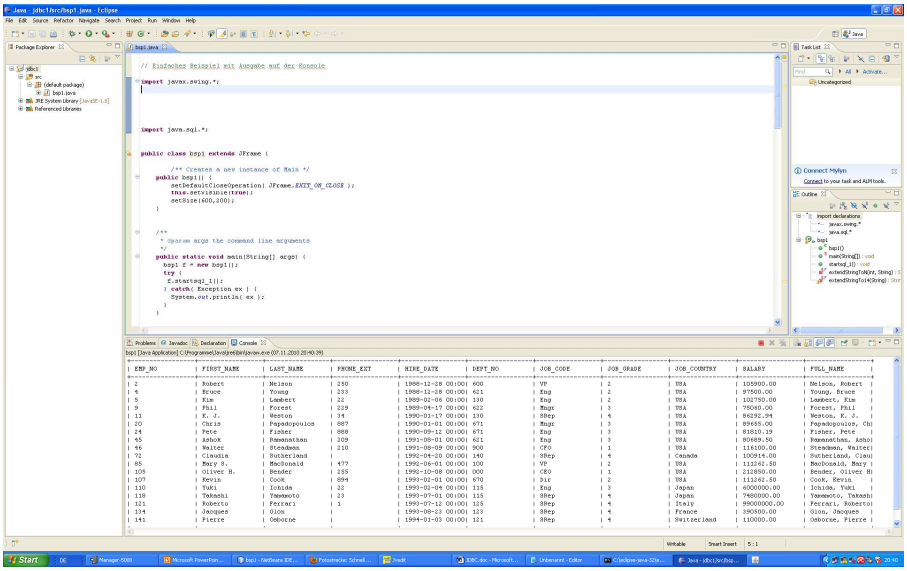
DBS-Treiber in Eclipse



DBS-Treiber in Eclipse



Ausführen in Eclipse

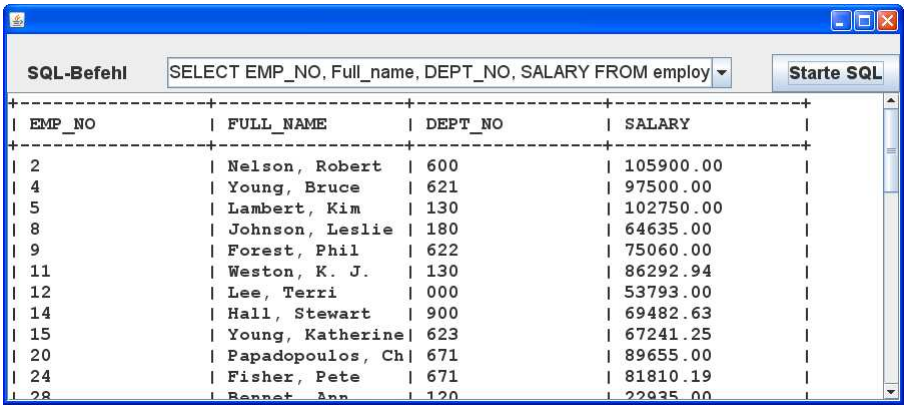


Ergebnis

EMP_NO	LAST_NAME	FIRST_NAME	DEPT_NO	SALARY
2	Nelson	Robert	600	105900.0
4	Young	Bruce	621	97500.0
5	Lambert	Kim	130	102750.0
9	Forest	Phil	622	75060.0
11	Weston	K. J.	130	86292.9375
20	Papadopoulos	Chris	671	89655.0
24	Fisher	Pete	671	81810.1875
45	Ramanathan	Ashok	621	80689.5
46	Steadman	Walter	900	116100.0
72	Sutherland	Claudia	140	100914.0
85	MacDonald	Mary S.	100	111262.5
105	Bender	Oliver H.	000	212850.0
107	Cook	Kevin	670	111262.5
110	Ichida	Yuki	115	6000000.0
118	Yamamoto	Takashi	115	7480000.0
121	Ferrari	Roberto	125	9.9E7
134	Glon	Jacques	123	390500.0
141	Osborne	Pierre	121	110000.0

- SELECT EMP_NO, Last_name, First_name, DEPT_NO, SALARY FROM " + sTable+" WHERE SALARY > 70000.0

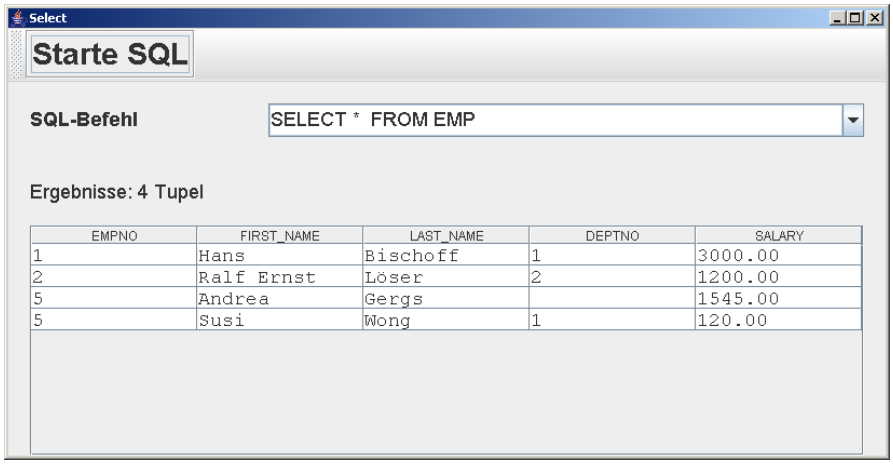
JDBC mit einem Editor



Beispiel mit test.gdb

- **Startfenster**
- **Fenster Select_Frame**
- **Fenster Insert_Frame**
- **Fenster Group_Frame**

Beispiel 3: JDBC mit Tabelle



EMPNO	FIRST_NAME	LAST_NAME	DEPTNO	SALARY
1	Hans	Bischoff	1	3000.00
2	Ralf Ernst	Löser	2	1200.00
5	Andrea	Gergs		1545.00
5	Susi	Wong	1	120.00

JDBC mit Tabelle

- **Zwei Vektoren**
  - **Zeilenvektor**
  - **Headervektor**
- **Änderungen beim Einlesen des SQL-Daten**
- **DataModel der Tabelle**
- **Updateproblem mit den Spalten**

JDBC mit Tabelle

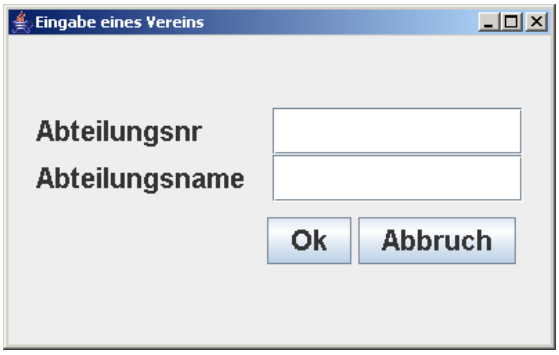
```
_tableHeader.removeAllElements();
for( i=0; i<_AnzCols; i++ ) {   // Schleife zählt von 1 bis N
     _tableHeader.add( rsmd.getColumnName( i+1 ) );
}

_tableRows.removeAllElements();
while( rs.next() ) {
   Vector Tupel = new Vector();  // neue Zeile
    for( i=0; i<_AnzCols; i++ ) {
        Tupel.add ( rs.getString( i+1 ) );
    }
   _tableRows.add( Tupel ); // Tupel zur Gesamtliste hinzufügen
}  // while
_AnzRows = _tableRows.size();
_tabelle.tableChanged(new TableModelEvent(
                _dataModel, TableModelEvent.HEADER_ROW));  // Spaltenproblem
_tabelle.updateUI();
```

JDBC mit Tabelle

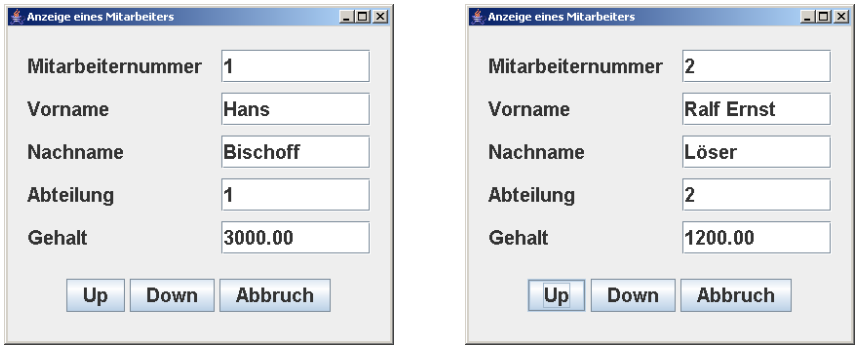
```
_dataModel = new AbstractTableModel() {
        public int getColumnCount() {
          return _tableHeader.size();
        }

        public int getRowCount() {
          return _tableRows.size();
        }

        public Object getValueAt(int row, int col) {
           Vector tupel = (Vector) _tableRows.elementAt(row);
            return tupel.elementAt(col);
        }

        public String getColumnName(int column) {
              return  _tableHeader.elementAt(column);
        }
};
```


Beispiel Insert_Frame

```
sql = "INSERT INTO DEPT (DEPTNO, DEPTNAME) "
            +"VALUES("+sDeptNr+ ", '"+ sDeptname+"');";
int anz = st.executeUpdate( sql );  // Returnwert Anzahl der betroffenen Datensätze
```



Beispiel Group_Frame

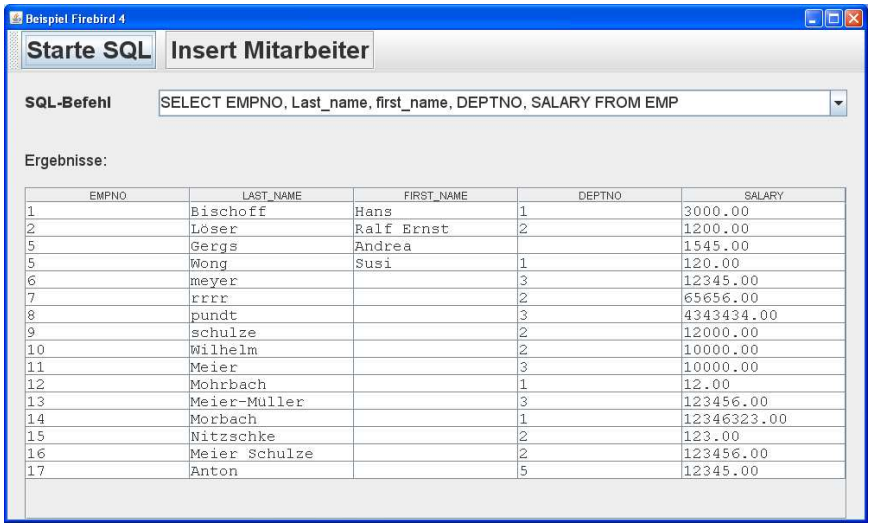


Beispiel 4: JDBC mit Tabelle und einem Dialog

- Neues Projekt erstellen
- Einbinden der jar-Datei in das Projekt
- Hauptprogramm erstellen
- Eintragen "import java.sql.*;"
- Event-Methode verknüpfen und schreiben

```
public void startsql()  throws Exception  {


}
```

Beispiel 4: JDBC mit Tabelle und einem Dialog



EMPNO	LAST_NAME	FIRST_NAME	DEPTNO	SALARY
1	Bischoff	Hans	1	3000.00
2	Löser	Ralf Ernst	2	1200.00
5	Gergs	Andrea		1545.00
5	Wong	Susi	1	120.00
6	meyer		3	12345.00
7	rrrr		2	65656.00
8	pundt		3	4343434.00
9	schulze		2	12000.00
10	Wilhelm		2	10000.00
11	Meier		3	10000.00
12	Mohrbach		1	12.00
13	Meier-Müller		3	123456.00
14	Morbach		1	12346323.00
15	Nitzschke		2	123.00
16	Meier Schulze		2	123456.00
17	Anton		5	12345.00

Beispiel 4: JDBC mit Tabelle und einem Dialog

JDBC mit Firebird (http://www.firebirdsql.org/)

- Hochskalierbare professionelle Datenbank (Borland)
- Sequenz / Generator
- PL-SQL
- computed by
  - Jahresgehalt=Monatsgehalt*12
  - Laenge=bis-von
- Constraints / Check / Default
- Mit Winforms / WPF nur eine DLL (kein Treiber notwendig)
- Unter Java wird die Datenbank benötigt (1 Minute Installation)

JDBC mit Firebird (http://www.firebirdsql.org/)

- Embedded Server requires that you use a special connection path string. It is similar to using the regular Firebird client library, however fbembed.dll on Windows and libfbembed.so on Linux are used.

  Here are the steps to make it work:
- 1. Unpack Firebird embedded .zip package in some directory. You need all the files, not just FBEMBED.DLL.
- 2. Set FIREBIRD environment variable to point to that directory.
- 3. Place the JAYBIRD.DLL in the PATH (by PATH, we mean the value of environment variable PATH)
- 4. Change the JDBC URL in your application, to something like this: jdbc:firebirdsql:embedded:/path/to/your/database.fdb
- 5. Start your Java application
  If this does not work, try specifying the following parameter to JVM: -Djava.library.path=<path_to_jaybird_dll>

Eigenschaften JDBC mit HsqlDB (hsqldb.org)

- 2. Varianten:
  - Datenbank nur im Hauptspeicher
  - Für große Datenbank existiert auch die Disk-Variante
- Sequenz
- PL-SQL
- Kein computed by
- Constraints / Check / Default
- 100 % Java
- Integriert in OpenOffice

Unterschide Firebird vs. HsqlDB

- Datenbank liegt nur im Hauptspeicher
- Sequence
  - CREATE TABLE SEQUENZ (PINDEX INTEGER NOT NULL, CONSTRAINT PK_SEQUENZ PRIMARY KEY (PINDEX));
  - CREATE SEQUENCE SEQ;
  - INSERT INTO Sequenz (pindex) Values(1);
  - "SELECT NEXT VALUE FOR "+seqname+" FROM Sequenz;";
- Kein computed by
- 100 % Java

JDBC mit Derby (http://db.apache.org/derby/)

- 2. Varianten:
  - Datenbank nur im Hauptspeicher
  - Für große Datenbank existiert auch die Disk-Variante
- Sequenz
- PL-SQL
- Kein computed by
- Constraints / Check / Default
- 100 % Java
- Integriert in OpenOffice

JDBC-Treiber

Firebird:

- sDbDrv = "org.firebirdsql.jdbc.FBDriver"
- sDbUrl="jdbc:firebirdsql:localhost/3050:D:\\dbs\test.fdb";
- jaybird-2.2.5.jar

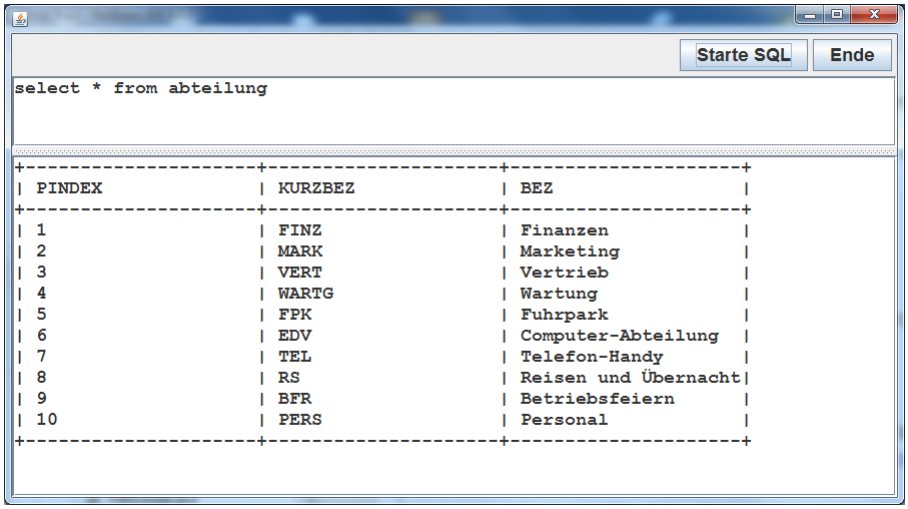
HdqlDB:

- sDbDrv="org.hsqldb.jdbcDriver";
- sDbUrl="jdbc:hsqldb:"+ databaseFile;
- hsqldb.jar

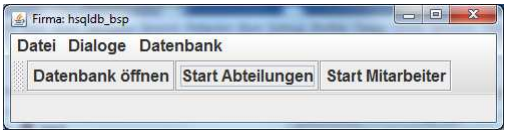
Apache Derby:

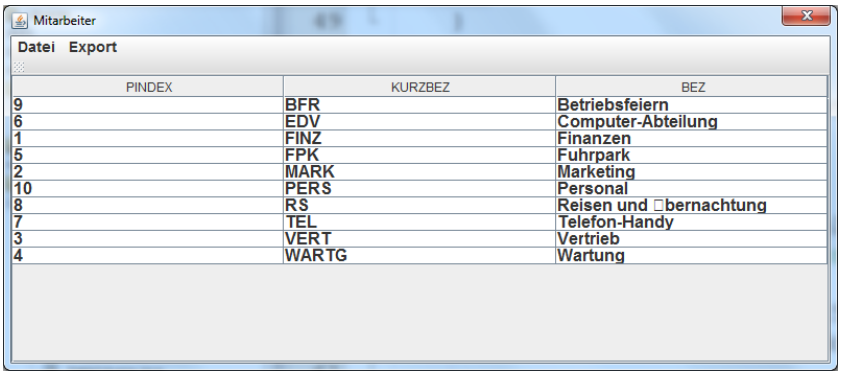
- sDbDrv="org.apache.derby.jdbc.ClientDriver";
- sDbUrl="jdbc:derby://localhost:1527/myDB;create=true;user=me;password =mine";

Hsql-Beispiele: Hsql_Bsp1.java

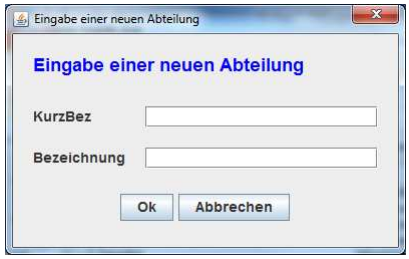


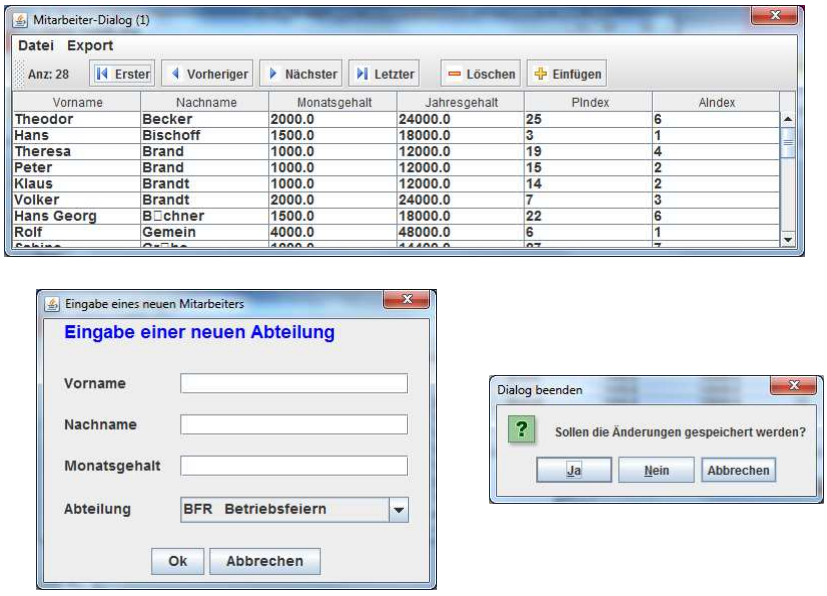
Hsql-Beispiele: Hsql_Bsp2.java

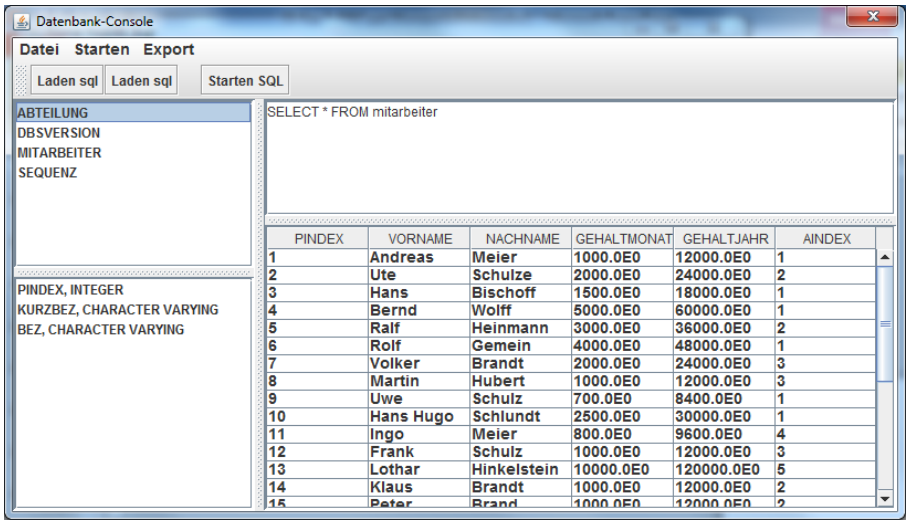




Hsql-Beispiele: Hsql_Bsp3.java




Mitarbeiter-Dialog (1)
Datei Export
Anz: 28
Erster
Vorheriger
Nächster
Letzter
Löschen
Einfügen
Vorname Nachname Monatsgehalt Jahresgehalt PIndex AIndex
Theodor Becker 2000.0 24000.0 25 6
Hans Bischoff 1500.0 18000.0 3 1
Theresa Brand 1000.0 12000.0 19 4
Peter Brand 1000.0 12000.0 15 2
Klaus Brandt 1000.0 12000.0 14 2
Volker Brandt 2000.0 24000.0 7 3
Hans Georg B□chner 1500.0 18000.0 22 6
Rolf Gemein 4000.0 48000.0 6 1
Eingabe eines neuen Mitarbeiters
Eingabe einer neuen Abteilung
Vorname
Nachname
Monatsgehalt
Abteilung
BFR Betriebsfeiern
Ok
Abbrechen
Dialog beenden
Sollen die Änderungen gespeichert werden?
Ja
Nein
Abbrechen


Datenbank-Console
Datei Starten Export
Laden sql
Laden sql
Starten SQL
ABTEILUNG
DBSVERSION
MITARBEITER
SEQUENZ
PINDEX, INTEGER
KURZBEZ, CHARACTER VARYING
BEZ, CHARACTER VARYING
SELECT * FROM mitarbeiter
PINDEX VORNAME NACHNAME GEHALTMONAT GEHALTJAHR AINDEX
1 Andreas Meier 1000.0E0 12000.0E0 1
2 Ute Schulze 2000.0E0 24000.0E0 2
3 Hans Bischoff 1500.0E0 18000.0E0 1
4 Bernd Wolff 5000.0E0 60000.0E0 1
5 Ralf Heinmann 3000.0E0 36000.0E0 2
6 Rolf Gemein 4000.0E0 48000.0E0 1
7 Volker Brandt 2000.0E0 24000.0E0 3
8 Martin Hubert 1000.0E0 12000.0E0 3
9 Uwe Schulz 700.0E0 8400.0E0 1
10 Hans Hugo Schlundt 2500.0E0 30000.0E0 1
11 Ingo Meier 800.0E0 9600.0E0 4
12 Frank Schulz 1000.0E0 12000.0E0 3
13 Lothar Hinkelstein 10000.0E0 120000.0E0 5
14 Klaus Brandt 1000.0E0 12000.0E0 2
15 Peter Brand 1000.0E0 12000.0E0 2