JDBC

SEW 4

DI Thomas Helml

SJ 2015/16

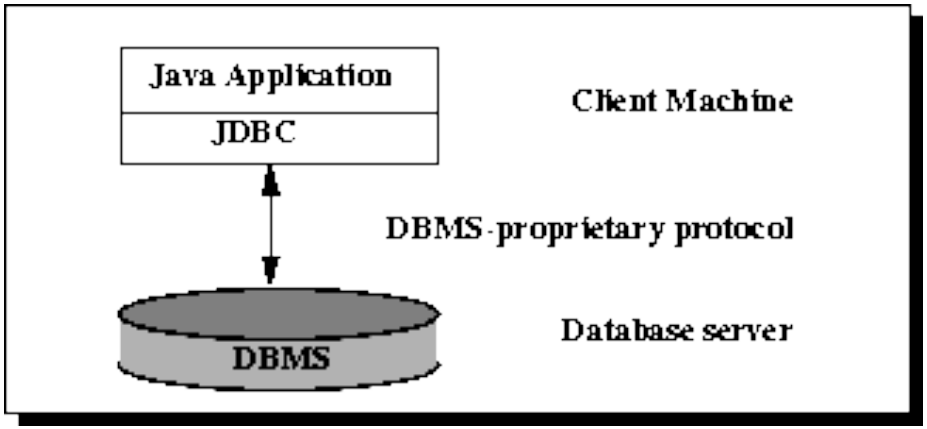
Inhaltsverzeichnis

JDBC

- Java DataBase Connectivity
  - Standard-Schnittstelle für Zugriff auf relationale DB mittels SQL und Java
  - Sammlung von Klassen und Interfaces
    - (Package: `java.sql`)
  - wird JDCBC verwendet: kein DB-spezifischer Code im Programmen
    - Abstraktionsschicht zw. Java und SQL

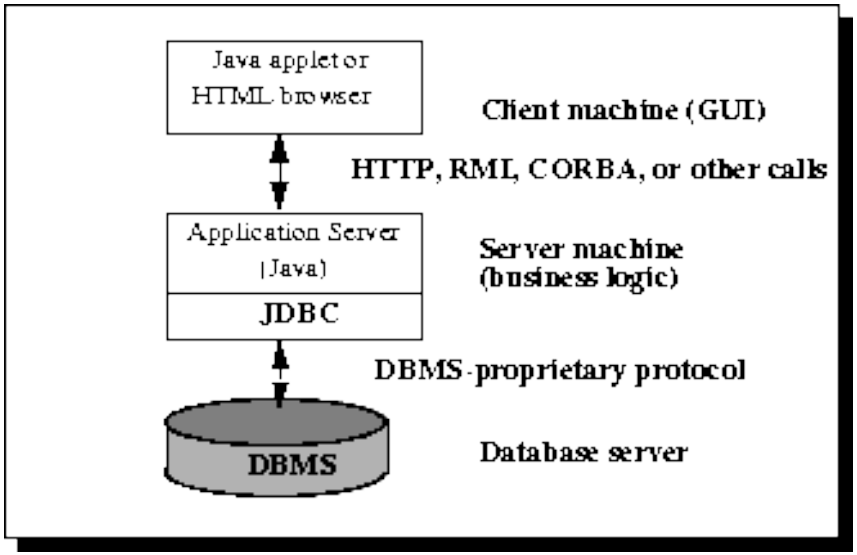
Anwendungsarchitektur

- 2-stufige Architektur
  - Client Programm greift direkt auf DB zu (Netzwerk oder lokal)



Anwendungsarchitektur

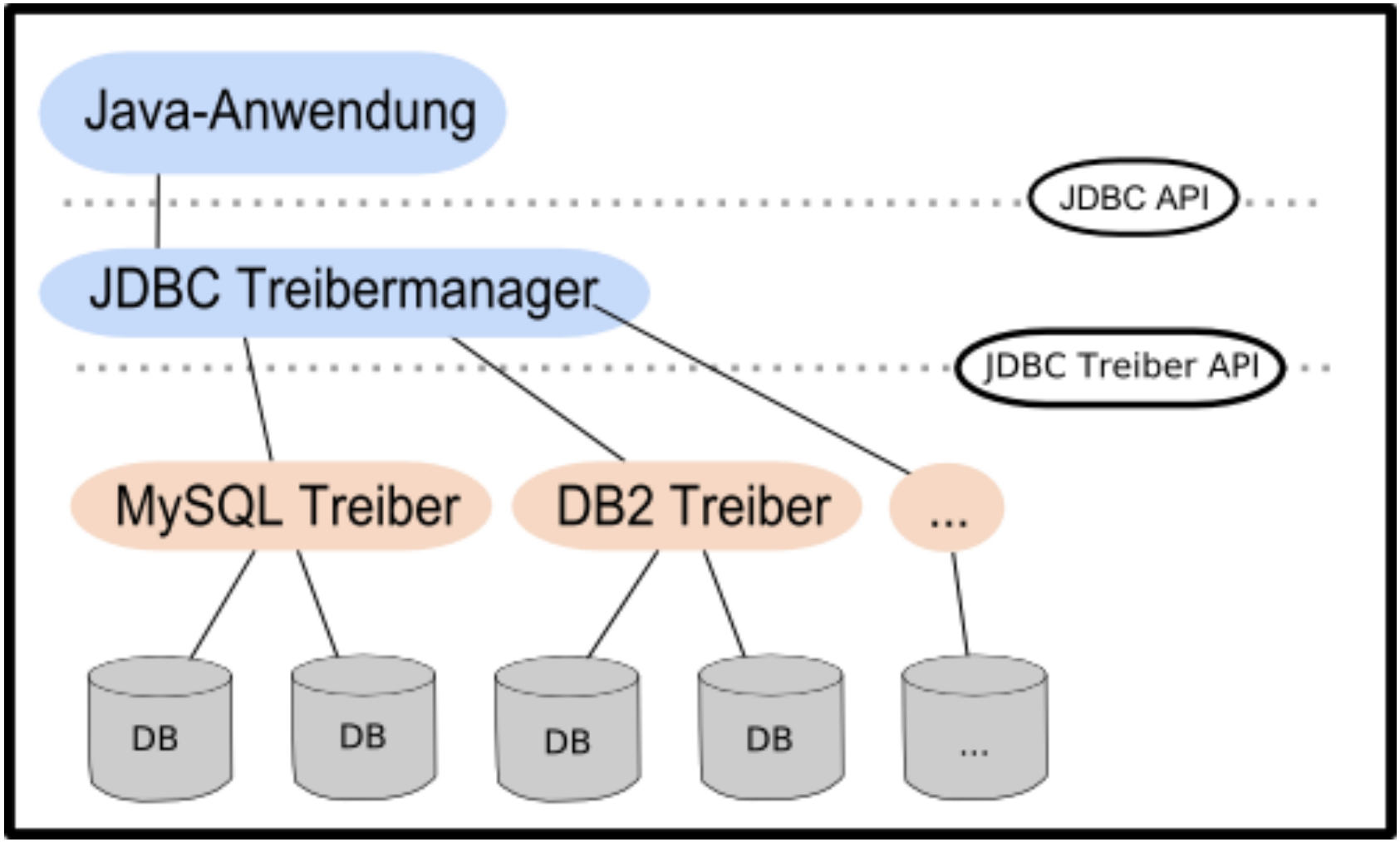
- 3-stufig:
  - Trennung: Anwendungslogik und Benutzeroberfläche bzw. Datenverwaltung
  - Client kommuniziert mit Applicationserver, der greift auf Datenbank zu



JDBC Treiber

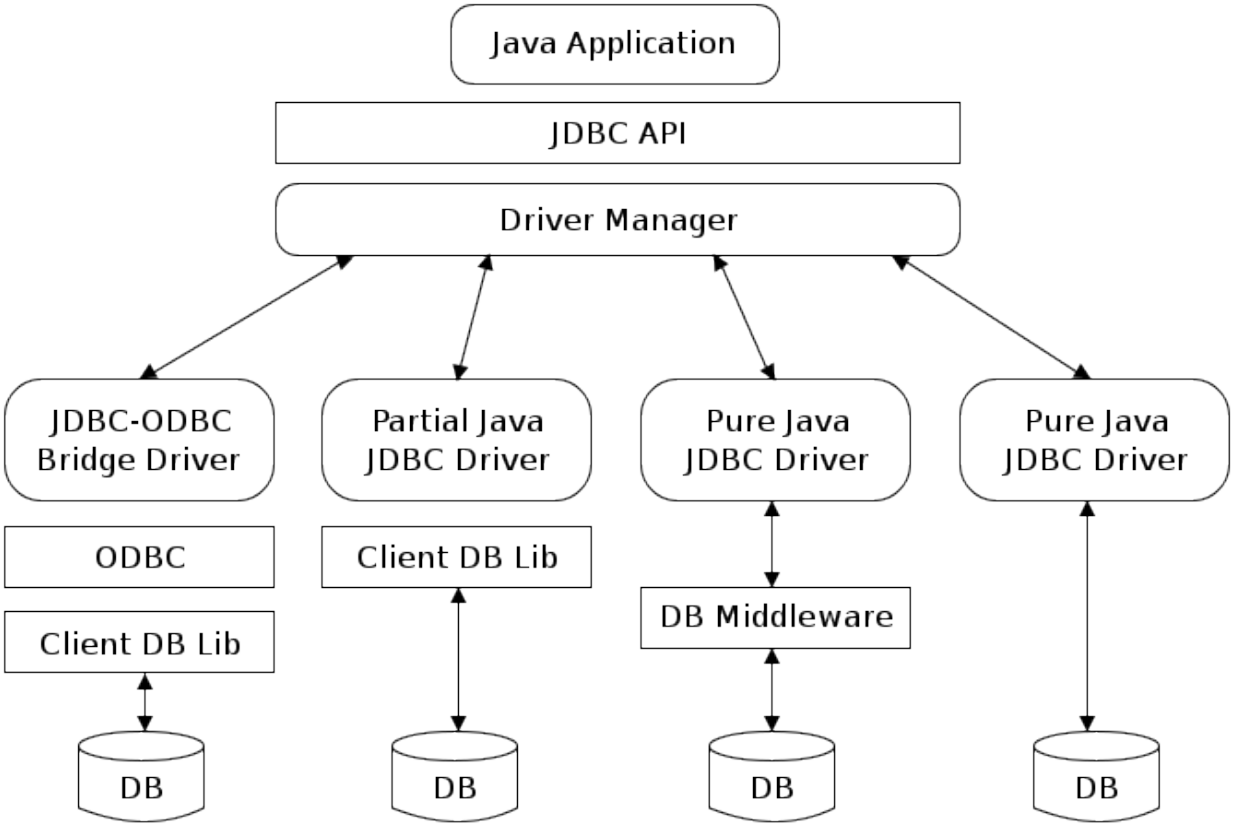
- jede DB hat herstellerabhängige Zugriffsschnittstelle
- JDBC-Treiber dient als Übersetzer auf diese Schnittstelle
- für jedes DBMS wird eigener Treiber benötigt

JDBC Treiber



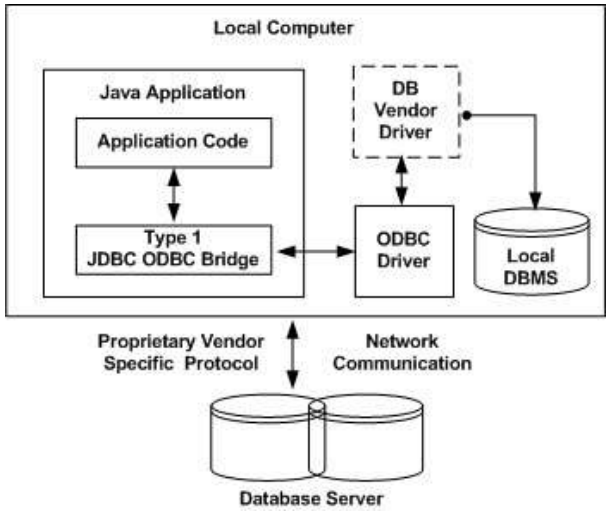
JDBC Treiber

- 4(3) Typen von Treiber:



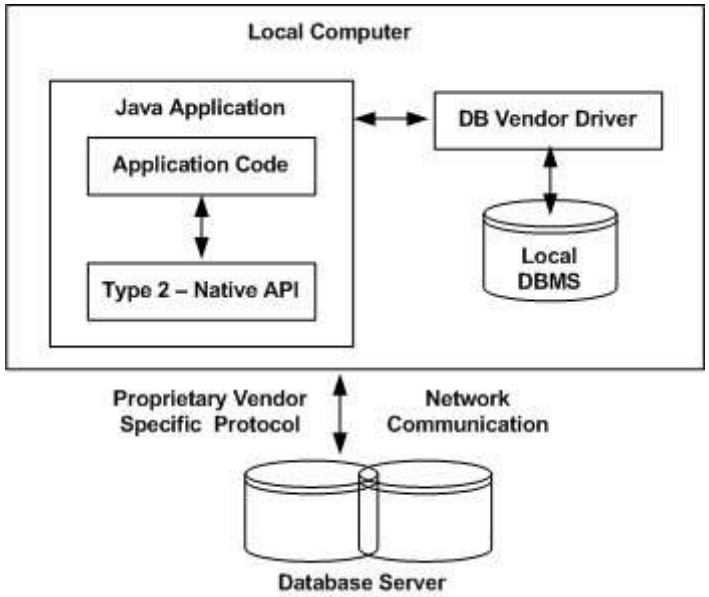
JDBC Treiber

- **Typ1: JDBC-ODBC-Bridge Treiber**
  - Treiber benutzen das ODBC (Open Database Connectivity) von Microsoft
  - Konvertierung von JDBC auf ODBC
  - ODBC Treiber müssen extra installiert werden
  - Paket: `oracle.jdbc.odbc`
  - `ab Java 8 entfernt`



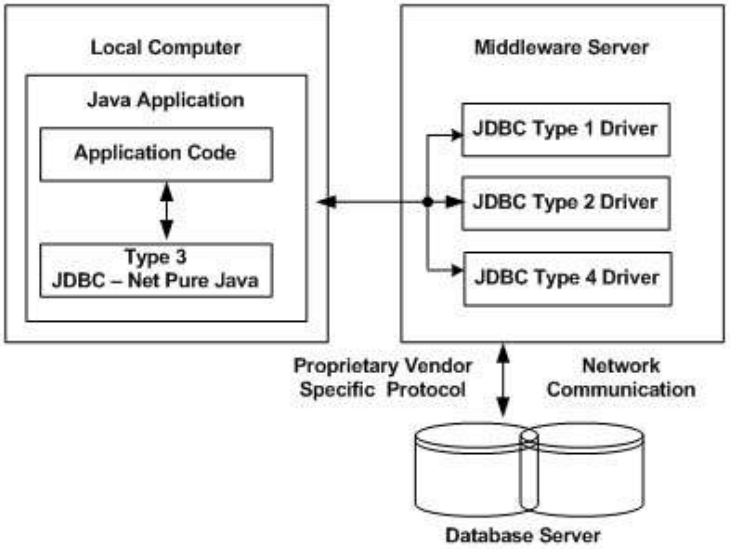
JDBC Treiber

- **Typ2: Native API Treiber**
  - Implementieren herstellerabhängige DB-Protokoll
  - 2 Bestandteile:
    - Java
    - plattformabhängiger Code (z.B. DLLs)
  - Anwendungen sind nur bedingt portierbar



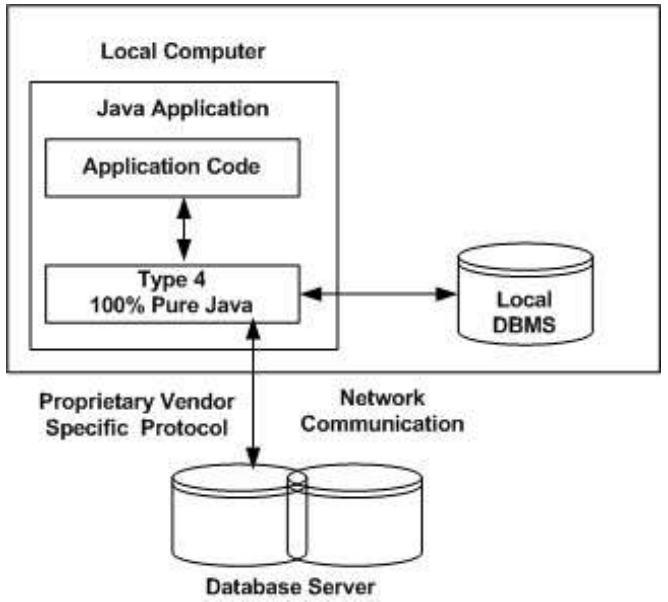
JDBC Treiber

- **Typ3: JDBC-Net Treiber**
  - komplett in Java realisiert
  - Zwischenschicht (Middleware) übernimmt Übersetzung von JDBC auf herstellerabhängiges DB-Protokoll
  - flexible Lösung: Wechsel des DBMS wird von Anwendung nichts bemerkt

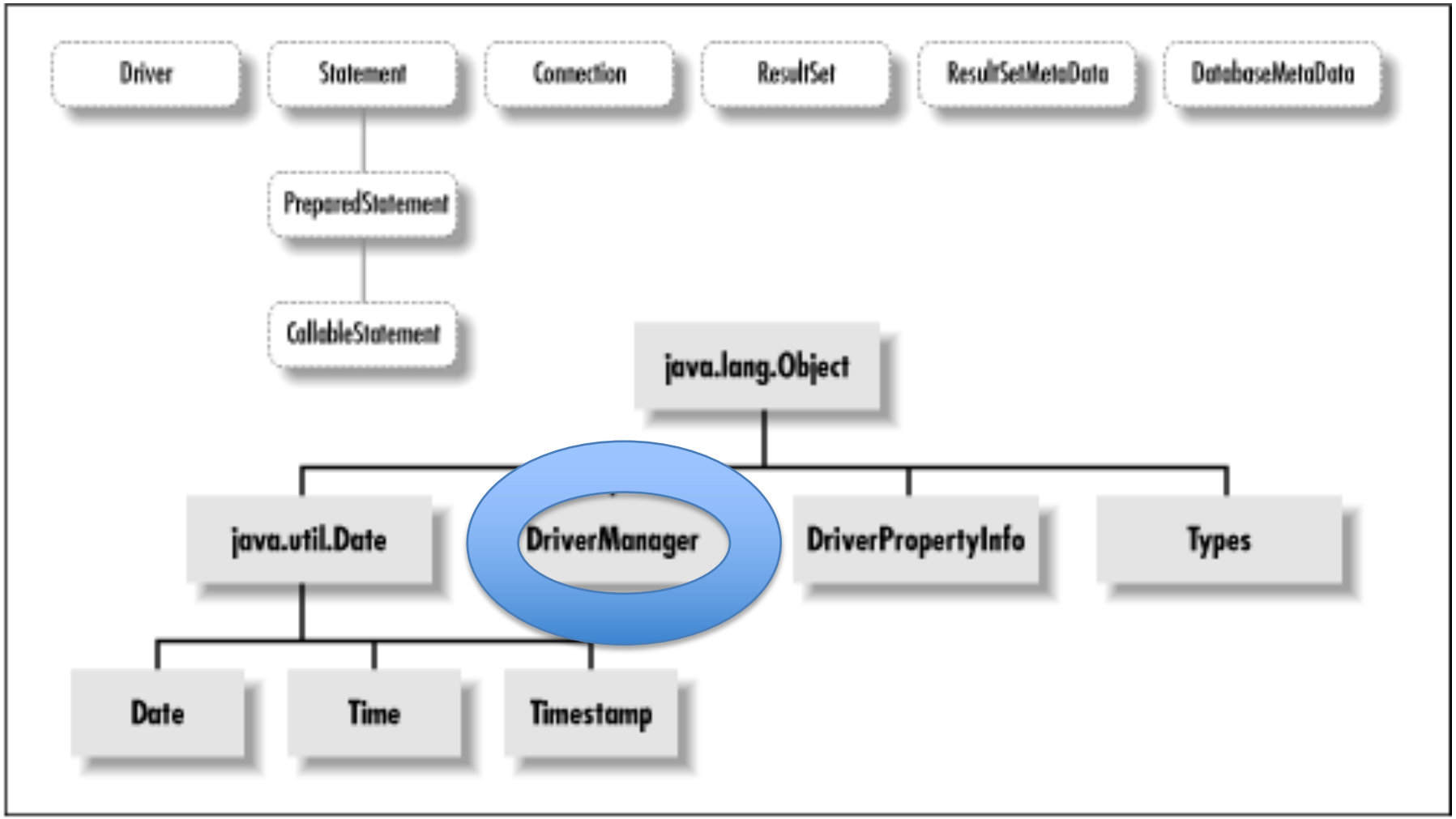


JDBC Treiber

- **Typ4: Native-Protocol Treiber**
  - Treiber komplett in Java
  - Implementieren DB Protokoll direkt
  - DBMS Hersteller liefern diese Treiber



JDBC API


Driver
Statement
Connection
ResultSet
ResultSetMetaData
DatabaseMetaData
PreparedStatement
CallableStatement
java.lang.Object
java.util.Date
DriverManager
DriverPropertyInfo
Types
Date
Time
Timestamp

JDBC API

- **Klassen:**
  - `DriverManager:`
    - Laden des JDBC-Treibers
    - Aufbau der Datenbankverbindung
  - `SQLException:`
    - Behandlung im Fehlerfall

JDBC API

- **Interfaces:**
  - `Connection`: repräsentiert eine DB-Verbindung
  - `Statement`: führt SQL Anweisungen über die DB-Verbindung aus
  - `ResultSet`: Methoden, um auf das Ergebnis der SQL-Abfrage zuzugreifen

JDBC Grundgerüst

Schritt 1: Treiber laden

```
try {
    Class.forName("org.hsqldb.jdbcDriver");
}
catch ( ClassNotFoundException e ) {
    System.err.println( "Keine Treiber-Klasse!" );
    return;
                                                    }
```

JDBC Grundgerüst

- **Schritt 2: Verbindungsaufbau**
  - externes Property File für Verbindungsdaten
  - „`dbconnect.properties`“
  - Vorteil: DBMS kann gewechselt werden kann, ohne Programmänderung

JDBC Grundgerüst

dbconnect.properties:

```
driver=org.hsqldb.jdbcDriver
url=jdbc:hsqldb:file:tutego
username=sa
password=

#Verbindungsdaten für MYSQL
#driver=com.mysql.jdbc.Driver
#url=jdbc:mysql://localhost:3306/tutego
#username=pc
#password=pc
```

Datenbanken werden über URL exakt identifiziert

jdbc:<subprotokoll>:<subname>

Subprotokoll: Art des verwendeten Treibers (z.B. odbc, mysql, ...)
Subname: ist die eigentliche Datenbank

JDBC Grundgerüst

- Code zum Laden der Eigenschaften aus Property-File:

```
try (
     FileInputStream in = new FileInputStream("dbconnect.properties");
)
{
     Properties prop = new Properties();

     // Properties laden
     prop.load(in);

     String driver = prop.getProperty("driver");
     String url = prop.getProperty("url");
     String user = prop.getProperty("user");
     String pwd = prop.getProperty("pwd");
}
catch (IOException e) {
     e.printStackTrace();
}
```

JDBC Grundgerüst

- Datenbankverbindung aufbauen:

```
Connection con =
    DriverManager.getConnection(url, user, pwd);
```

- Zugriff auf DB erfolgt über `Connection`-Objekt!
- Info über DBMS abfragen:

```
DatabaseMetaData getMetaData() throws SQLException
```

Aufgabe

- Laden Sie hsqldb 2.3.2 herunter:
  - http://sourceforge.net/projects/hsqldb/
- HSQL DB Manager starten:
  - bin/runManagerSwing.bat
- Options - Insert Test Data

Aufgabe

a) Erstellen Sie ein Programm „PropertyTest“

b) Erstellen Sie ein Property File für HSQL und erweitern Sie es für Ihre Postgres Datenbank!

c) Bauen Sie eine Verbindung zur HSQL und Postgres DB auf und geben Sie die Metaeigenschaften

- URL der DB
- Name des DB Users
- Produktnamen des DBMS
- Version der DBMS
- Namen des JDBC Treibers
- Version des JDBC Treibers

am Bildschirm aus!

Aufgabe

- Hsqldb in Eclipse einbinden:
  - Projekteigenschaften
  - Java Buid Path – Add External JARs