JDBC

- Java DataBase Connectivity
- Sammlung von Klassen und Interfaces zur Arbeit mit Datenbanken auf Basis von SQL
- Die Klassen befinden sich im Package java.sql.
- Um mit einer Datenbank zu kommunizieren bedarf es eines Jdbc Drivers, der die Klasse driver.class implementiert.
- Der jdbc-Driver enthält die wesentliche Funktionalität zur Kommunikation mit einer Datenbank

Hauptschritte einer JDBC-App

- Laden und registrieren eines JDBC-Drivers, passend zum Datenbankserver (zB. mysql)
- Connection object vom Drivermanager erzeugen lassen, dazu Logindaten und URL übergeben.
- Mit Hilfe der Connection ein oder mehrere Statement Objekte erzeugen.
- Sql Arbeitsschritt mit Statement und Querystring zur Ausführung bringen, Ergebnis ist ein int-Wert oder ein Resultset.
- Beenden mit Schließen der Connection (close).

JDBC Driver (4 Typen)

- Typ 1: JDBC-ODBC Bridge

  Benutzt ODBC Treiber, ODBC muss auf jeder client Maschine verfügbar sein. Die JDBC Aufrufe werden in ODBC Aufrufe umgewandelt (langsam). War vor allem nach Veröffentlichung von jdbc bedeutsam.

- Typ 2: native API partly Java driver

  Benutzt das API der jeweiligen Datenbank, die JDBC Aufrufe werden in API Aufrufe umgewandelt.

- Typ 3: JDBC-Net pure Java Driver

  Transformiert die JDBC Aufrufe in ein datenbankunabhängiges Netzprotokoll, das durch einen geeigneten Server dann interpretiert wird. Dieser Server wird auch Middleware genannt.

- Typ 4: Native protocol pure Java Driver

  Transformiert die JDBC Aufrufe in ein von der Datenbank direkt unterstütztes Netzprotokoll. Ist heute der übliche Standard.

Bereitstellen des Treibers

- Meist liegt der Treiber in Form eines jar-Files vor.
- Nutzung der java-Option -cp (-classpath) oder Setzen der Umgebungsvariable CLASSPATH um das Teiberpackage nutzen zu können. Die Option -cp muss unmittelbar hinter java oder javac angegeben werden.

```
javac –cp .:driverxyz.jar myapp.java

java –cp .:driverxyz.jar myapp
```

- **Oder**

```
export CLASSPATH=.:driverxyz.jar
```

Ggf. Pfad mit angeben, wo Sich die .jar Datei befindet

Der Classpath muss Das aktuelle Verzeichnis enthalten

Laden des Treibers

- Einkompilieren des Treibers (eher unüblich)

```
// org.gjt.mm.mysql.Driver x=
    new org.gjt.mm.mysql.Driver();
```

- Laden des Treibers zur Laufzeit (üblich)

```
Class.forName("org.gjt.mm.mysql.Driver");
```

- Erklärung: mit Hilfe der statischen Methode forName der Klasse Class ist es möglich, in ein laufendes Javaprogramm Klassen zu (nachzu-) laden.

Benutzen des jdbc Driver

- Die Benutzung erfolgt über den Drivermanager durch Aufruf der Methode

  `Drivermanager.getConnection(url,"user","password");`

```
import java.sql.*;

public class jdbcTest
{
 public static void main(String args[])
 {
  Connection con;
  try
  {
   Class.forName("org.gjt.mm.mysql.Driver");
   String url = "jdbc:mysql://localhost:3306/ADDR";
   con = DriverManager.getConnection(url,"beck","spass");
   . . .
```

- Packagestruktur kann aus der Doku oder dem jar-File ermittelt werden (jar tvf myLovelyJarFile.jar).
- Protokoll der URL aus Doku oder Beispielprogrammen ermitteln.(http://www.developer.com/article.php/3417381#Discussion%20and%20Sample%20Programs)
- Bei der Variante des Ladens des Driver über Class.forName stellt sich dem aufmerksamen Betrachter die Frage, woher kommt die Instanz des Drivers und wie erhält der Drivermanager Kenntis von selbiger?
- Um dieses Rätsel zu lösen, hilft ein Blick in die Quellen eines jdbc Drivers.

```
//**************************************************
//The Driver class is internally known as jdbcMysqlDriver.
//**************************************************

public final class jdbcMysqlDriver implements Driver
{
/** Self instantiation */
static { new jdbcMysqlDriver(); }

/** Keep track of connections just in case we are interested.
*/
static Vector connections;

/** Locate defaults */
static jdbcMysqlBase mysql;
```

Statischer Initialisierer,
wird automatisch nach dem Laden
der Klasse aufgerufen.
Instanziert den Driver

- Das Registrieren des Drivers erfolgt nun im Constructor

```
public jdbcMysqlDriver()
{
 this.connections = new Vector();
 this.mysql = new jdbcMysqlBase(this);
 try
 {
   DriverManager.registerDriver(this);
 }
 catch(SQLException se)
 {
   System.out.println(
      "Error registering twz1.jdbc.mysql Driver\n" + se);
 }
}
```

Datenbank für Programmbeispiele

AddrB:

Field	Type	Null	Key	Default	Extra
PersID	int(11)		PRI	0	
Name	varchar(39)	YES			
FName	varchar(30)	YES			
Street	varchar(39)	YES			
ZIP	varchar(12)	YES			
Town	varchar(30)	YES			
Note	varchar(30)	YES			
LastUpdate	timestamp(10)	YES			

TeleB:

Field	Type	Null	Key	Default	Extra
PhoneID	int(11)		PRI	0	auto_increment
PersID	int(11)			0	
Note	varchar(30)	YES			
Phone	varchar(15)	YES			
LastUpdate	timestamp(10)	YES			

Test der Connection

- Ermitteln von Metadaten zur Datenbank, ist dies erfolgreich, scheint die Verbindung zum DBMS und zur Datenbank zu funktionieren.

```
DatabaseMetaData dmd=con.getMetaData();
```

- Nun kann man mit dem DMS schon kommunizieren:

```
System.out.println("URL: "+dmd.getURL());

System.out.println("Product: "
+dmd.getDatabaseProductName()+" Vers."
+dmd.getDatabaseProductVersion() );

System.out.println("Driver : "
+dmd.getDriverName()+" Vers."
+dmd.getDriverVersion() );
```

Komplettes Beispiel

```
import java.sql.*;
public class jdbcTest
{
 public static void main(String args[])
 {
  Connection con;
  ResultSet  Rs;
  try
  {
   Class.forName("org.gjt.mm.mysql.Driver");
   String url = "jdbc:mysql://localhost:3306/ADDR";
   System.out.println("Driver:"+DriverManager.getDriver(url));
   con = DriverManager.getConnection(url,"beck","spass");
   DatabaseMetaData dmd=con.getMetaData();
   if (con!=null)
   {
    System.out.println("URL: "+dmd.getURL());
    System.out.println(„Product: "+dmd.getDatabaseProductName()
              +" Vers."+dmd.getDatabaseProductVersion() );
    System.out.println("Driver : "+dmd.getDriverName()
              +" Vers."+dmd.getDriverVersion() );
   }
  }
  catch(Exception e){System.out.println(e);e.printStackTrace();}
 }
}
```

Passen Sie JDBC-Driver und URL an Ihre Datenbank an.

Kommunikation mit Datenbanken

- Zur Kommunikation wird SQL genutzt, ggf. im Dialekt des verwendeten DBMSs.
- Prinzipiell wird zwischen zwei Arten von SQL-Statements unterschieden.
  - Statements, die einen ganzzahligen Wert liefern
  - Statements, die ein Resultset liefern
- Um SQL-Statements absetzen zu können, bedarf es eines Objektes der Klasse Statement:

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

- In einem QueryString, wird das SQL-Statement formuliert:

```
String Query="Select * from AddrB where Town='Buschhausen'"
```

execute

- Die Klasse Statement stellt mehrere execute... Methoden bereit, die wichtigsten sind:
  - executeQuery

    ```
    ResultSet executeQuery(String sql);
    ```

  - executeUpdate

    ```
    int executeUpdate(String sql);
    ```

- Entsprechend des zu erwartenden Ergebnisses ist die Methode executeUpdate oder executeQuery aufzurufen.

Resultset

- Liefern SQL-Statements eine Liste von Ergebnissen, so werden diese in Form eines ResultSet zurückgegeben.
- Das ResultSet enthält in Form einer Tabelle die Resultate einer Datenbankanfrage.
- Die Zeilen werden über einen „Cursor“ adressiert. In Java navigiert man mit den Methode next, previous, relative oder absolute.
- Die erste Zeile hat dabei den Index 1!
- Typischerweise navigiert man sequenziell durch das ResultSet mit next, dabei indiziert der erste Aufruf von next die erste Zeile.

```
Rs=stmnt.executeQuery(Query);
while(Rs.next())
{
 System.out.printf("%-10s %-10s %-15s %05d,%s\n",
                   Rs.getString(2),
                   Rs.getString(3),
                   Rs.getString(4),
                   Rs.getInt(5),
                   Rs.getString(6));
}
```

Auf die Spalten des Resultset wird über spezielle Methoden zugegriffen (getInt, getString,...), je nach zu erwartendem Wert.

- Die Indizierung der Spalten beginnt mit ebenfalls mit 1!
- Anstelle des Spaltenindex kann auch der Spaltenname als String angegeben werden

MS SQL

- die Daten zum MS SQL-Server:
- Server: 141.56.2.45
- Port: 1433

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
String  url="jdbc:sqlserver://141.56.2.45:1433;databaseName=privbeck;";
System.out.println("Driver:"+DriverManager.getDriver(url));
con = DriverManager.getConnection(url,"beck","spass");
DatabaseMetaData dmd=con.getMetaData();
```

Instanz: MSSQL2012 wird nicht angegeben

JDBC-Treiber und Informationen zum Erstellen des Verbindungsstrings finden Sie hier:

https://msdn.microsoft.com/de-de/library/mt484311%28v=sql.110%29.aspx