

JDBC Kurzanleitung

Aus: Dehnhardt, Wolfgang: Anwendungsprogrammierung mit JDBC. © Hanser 1999, ISBN 3-446-21265-5

Treiber laden und registrieren

```
Class.forName("package.TreiberKlasse");      z.B.  
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Verbinden

```
Connection c =  
    DriverManager.getConnection(url, login, password);  
url = "protokoll:subprotokoll:datenbank";      oder  
url = "protokoll:subprotokoll://ip-adresse:portnr/datenbank";  
z.B.  
url = "jdbc:odbc:Kurse";                        oder  
url = "jdbc:msql://www.beispiele.de:1112/Kurse";
```

Statement und ResultSet

```
Statement s = c.createStatement();  
ResultSet rs = s.executeQuery("sqlAusdruck");  
int geänderteZeilen = s.executeUpdate("sqlAusdruck");
```

```
if (s.execute("sqlAusdruck")) // true:  
    ResultSet rs = s.getResultSet(); // ResultSet  
else // false:  
    int geänderteZeilen = s.getUpdateCount(); // Zähler
```

Vorbereitete Anweisungen

```
PreparedStatement ps = c.prepareStatement  
    ("UPDATE tabelle SET spalte = ? WHERE nr = ?");  
(Fragezeichen werden mit 1 beginnend durchgezählt.)  
ps.setString(1, "Robusta"); // erstes ?  
ps.setInt(2, 27); // zweites ?  
ResultSet rs = ps.executeQuery();  
int geänderteZeilen = ps.executeUpdate();  
boolean b = ps.execute();
```

Gespeicherte Prozeduren

```
CallableStatement cs = c.prepareCall  
    ("{call meineProzedur(?, ? ...)}");  
CallableStatement cs = c.prepareCall  
    ("{? = call meineFunktion(?, ? ...)}");  
(Fragezeichen werden mit 1 beginnend durchgezählt.)  
cs.registerOutParameter(paramIndex, Types.datenTyp)  
cs.setTyp(paramIndex, wert) (Typ = Int, Float, String, ...)  
ResultSet rs = cs.executeQuery();  
int geänderteZeilen = cs.executeUpdate();  
boolean b = cs.execute();
```

Auswertung

```
while (rs.next()) {  
    String sp1 = rs.getString("spalte1Name"); // Spalte 1  
    int sp2 = rs.getInt(2); // Spalte 2  
}
```

Programmbeispiele zum Buch:

<http://www.inf.uni-hohenheim.de/buch/jdbc>

Java: JDK, JDBC, Dokumente:

<http://java.sun.com>, z.B.
[http://java.sun.com/products/jdk/1.x\(x=1,2\)](http://java.sun.com/products/jdk/1.x(x=1,2)) und <http://java.sun.com/products/jdbc>

Referenzen:

- Hamilton/Cattell/Fisher: JDBC. Datenbankzugriff mit Java. © Addison-Wesley 1998, ISBN 3-8273-1306-6 (JDBC Version 1)
- White/Fisher/Cattell/Hamilton/Hapner: JDBC API Tutorial and Reference: Universal Data Access for the Java 2 Platform. © Addison-Wesley 1999, 2. Auflage, ISBN 0-201-43328-1 (JDBC Version 2)

Beispiel (mit Metadaten, s.u.)

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  
Connection c = DriverManager.getConnection  
    ("jdbc:odbc:Kurse", "ich", "");  
Statement s = c.createStatement();  
if (s.execute("SELECT * FROM Personen")) {  
    ResultSet rs = s.getResultSet();  
    ResultSetMetaData rsmd = rs.getMetaData();  
    int max = rsmd.getColumnCount();  
    for (int i = 1; i <= max; i++)  
        System.out.println(rsmd.getColumnName(i));  
    while(rs.next())  
        for (int i = 1; i <= max; i++)  
            System.out.println(rs.getString(i));  
}  
else  
    System.out.println(s.getUpdateCount());
```

Metadaten

```
DatabaseMetaData dbmd = c.getMetaData();  
ResultSetMetaData rsmd = rs.getMetaData();
```

JDBC-Typen

Siehe JDBC-Klasse `java.sql.Types`
und evtl. JDBC-Implementierungen `XxxTypes`
(z.B. `oracle.jdbc.driver.OracleTypes`)
sowie auf der Rückseite

SQL

```
SELECT spaltenListe  
FROM tabellenListe  
[WHERE bedingungen]  
[GROUP BY spalte  
    [HAVING bedingungen] ]  
[ORDER BY spaltenListe [ASC | DESC] ]
```

```
UPDATE tabelle  
SET spaltenWerteListe  
[WHERE bedingungen]
```

```
INSERT INTO tabelle [(spaltenListe)]  
VALUES (werteListe)
```

```
INSERT INTO tabelle [(spaltenListe)]  
SELECT spaltenListe  
FROM tabellenListe ...
```

```
DELETE FROM tabelle  
[WHERE bedingungen]
```

	TINYINT	SMALLINT	INTEGER	BIGINT	REAL	FLOAT	DOUBLE	DECIMAL	NUMERIC	BIT	CHAR	VARCHAR	LONGVARCHAR	BINARY	VARBINARY	LONGVARBINARY	DATE	TIME	TIMESTAMP	
String	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
java.math.BigDecimal	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Boolean	x	x	x	x	x	x	x	x	x	x	x	x	x							
Integer	x	x	x	x	x	x	x	x	x	x	x	x	x							
Long	x	x	x	x	x	x	x	x	x	x	x	x	x							
Float	x	x	x	x	x	x	x	x	x	x	x	x	x							
Double	x	x	x	x	x	x	x	x	x	x	x	x	x							
byte[]														x	x	x				
java.sql.Date											x	x	x				x			x
java.sql.Time											x	x	x					x		x
Java.sql.Timestamp											x	x	x					x		x

Mit *PreparedStatement.setObject()* mögliche Umwandlungen von Java- in JDBC/SQL-Typen

"x" bedeutet, daß mit der Methode *setObject()* aus der Klasse *PreparedStatement* der links angegebene Java-Typ in die entsprechenden JDBC/SQL-Typen umgewandelt werden kann.

	TINYINT	SMALLINT	INTEGER	BIGINT	REAL	FLOAT	DOUBLE	DECIMAL	NUMERIC	BIT	CHAR	VARCHAR	LONGVARCHAR	BINARY	VARBINARY	LONGVARBINARY	DATE	TIME	TIMESTAMP	
getBytes()	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
getDate()											x	x	x				x			x
getTime()											x	x	x					x		x
getTimeStamp()											x	x	x					x		x
getAsciiStream()											x	x	x	x	x	x				
getUnicodeStream()											x	x	x	x	x	x				
getBinaryStream()														x	x	x				
getObject()	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Mit *ResultSet.getXxx()*-Methoden abrufbare Datentypen

"x" heißt, die in der linken Spalte angegebene Methode kann den entsprechenden JDBC-Typ abrufen; "x" bedeutet, daß für einen Typ diese Methode zum Abruf empfohlen wird.

JDBC-Typ	⇒	Java-Typ	⇒	JDBC-Typ
CHAR		String		VARCHAR bzw. LONGVARCHAR
VARCHAR				
LONGVARCHAR				
NUMERIC		java.math.BigDecimal		NUMERIC
DECIMAL				
BIT		Boolean	boolean	BIT
TINYINT		Integer		
SMALLINT			int	INTEGER
INTEGER				
BIGINT		Long	long	BIGINT
REAL		Float	float	REAL
FLOAT		Double		
DOUBLE			double	DOUBLE
BINARY		byte[]		VARBINARY bzw. LONGVARBINARY
VARBINARY				
LONGVARBINARY				
DATE		java.sql.Date		DATE
TIME		java.sql.Time		TIME
TIMESTAMP		java.sql.Timestamp		TIMESTAMP

Abbildung von JDBC/SQL-Typen auf Klassen (String, Integer etc.) bzw. primitive Typen (int, double etc.) in Java und umgekehrt