



# JDBC Grundgerüst

---

- ⑦ Basisschnittstelle für alle SQL-Anweisungen:

```
java.sql.Statement
```

- ⑦ Statement Object erzeugen:

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

- ⑦ SQL-Anweisungen ausführen

```
ResultSet rs = stmt.executeQuery("SELECT * FROM CUSTOMER" );
```

- ⑦ SQL-Anweisung enthält kein abschließendes Semikolon

- ⑦ unterschiedlich von DBMS zu DBMS
- ⑦ Treiber ergänzt dies bei Bedarf



# JDBC Grundgerüst

---

- ④ Ergebnis der SQL Abfrage (Tabelle): `ResultSet`
- ④ Ergebniscursor (=Position in Ergebnismenge)
  - ④ Zeilenweise vorwärts mittels `.next()`
  - ④ Anfang: `.first()`, Ende: `.last()`
- ④ Spaltenzugriff über `getXXX()` Methoden
  - ④ Spaltenindex bzw. Spaltenname als Parameter



# JDBC Grundgerüst

---

- ④ Entsprechend den Java-Typen gibt es passende getter-Methoden in der Klasse `ResultSet`:
  - ④ `byte getByte(...)`
  - ④ `short getShort(...)`
  - ④ `int getInt(...)`
  - ④ ...



# JDBC Grundgerüst

---

## ① Beispiel 1:

```
ResultSet rs = stmt.executeQuery( "SELECT * FROM CUSTOMER" );
while ( rs.next() )
    System.out.printf( "%s, %s %s, %s %s \n",
        rs.getString(1),
        rs.getString(2),
        rs.getString(3),
        rs.getString(4),
        rs.getString(5)
    );
```



# JDBC Grundgerüst

---

## ① Beispiel 2:

```
rs = stmt.executeQuery( "SELECT LASTNAME,  
                        CITY FROM Customer" );  
  
while ( rs.next() )  
    System.out.printf( "%s, %s  \n",  
        rs.getString(1),  
        rs.getString(2)  
    );
```



# JDBC Grundgerüst

---

## ④ Beispiel 3:

```
rs = stmt.executeQuery( "SELECT LASTNAME,  
                        CITY FROM Customer" );  
  
while ( rs.next() )  
    System.out.printf( "%s \n",  
                      rs.getString( "LASTNAME" ) );
```



# JDBC Grundgerüst

---

- ① Freigabe der DB Ressourcen
  - ① ResultSet
  - ① Statement
  - ① Connection
- ① mit `.close()` schließen!!



# JDBC Grundgerüst

---

## ④ Datenbankänderungen

- ④ Änderungen der DB-Struktur (z.B. CREATE TABLE)
- ④ Änderungsbefehle (INSERT, UPDATE, DELETE)

```
int executeUpdate(String sql)  
throws SQLException
```

## ④ Rückgabewert bei INSERT, UPDATE, DELETE

- ④ Anzahl der geänderten Datensätze





# JDBC Datentypen

---

- ④ SQL arbeitet mit anderen Datentypen als Java
- ④ => eigene JDBC-Datentypen zur Typkonvertierung definiert
- ④ zu finden in: `java.sql.Types`



# JDBC Datentypen

---

JDBC Typ	Java Typ	Java Object-Typ
TINYINT	byte	java.lang.Byte
SMALLINT	short	java.lang.Short
INTEGER	int	java.lang.Integer
BIGINT	long	java.lang.Long
REAL	double	java.lang.Double
FLOAT	double	java.lang.Double
DOUBLE	double	java.lang.Double



# JDBC Datentypen

---

JDBC Typ	Java Typ	Java Object-Typ
DECIMAL	<code>java.math.BigDecimal</code>	<code>java.math.BigDecimal</code>
NUMERIC	<code>java.math.BigDecimal</code>	<code>java.math.BigDecimal</code>
DATE	<code>java.sql.Date</code>	<code>java.sql.Date</code>
TIME	<code>java.sql.Time</code>	<code>java.sql.Time</code>
TIMESTAMP	<code>java.sql.Timestamp</code>	<code>java.sql.Timestamp</code>



# JDBC Datentypen

---

JDBC Typ	Java Typ	Java Object-Typ
CHAR	<code>java.lang.String</code>	<code>java.lang.String</code>
VARCHAR	<code>java.lang.String</code>	<code>java.lang.String</code>
LONGVARCHAR	<code>java.lang.String</code>	<code>java.lang.String</code>
BIT	<code>boolean</code>	<code>java.lang.Boolean</code>
BINARY	<code>byte[]</code>	<code>byte[]</code>
VARBINARY	<code>byte[]</code>	<code>byte[]</code>
LONGVARBINARY	<code>byte[]</code>	<code>byte[]</code>



# ÜBUNG

---

- ① Übung 1 – JDBC Datentypen
- ① Übung 2 – JDBC Grundgerüst