Java:
MySQL-Anbindung mit JDBC

Vorarbeiten

Wir brauchen:

- MySQL-Server
- JDBC-Treiber
- (Import java.sql.*)

Vorarbeiten

MySQL-Server

in unserem Falle: WAMP (= Apache)

(runterladen, installieren, starten)

Vorarbeiten

JDBC (Java Database Connectivity)-Treiber

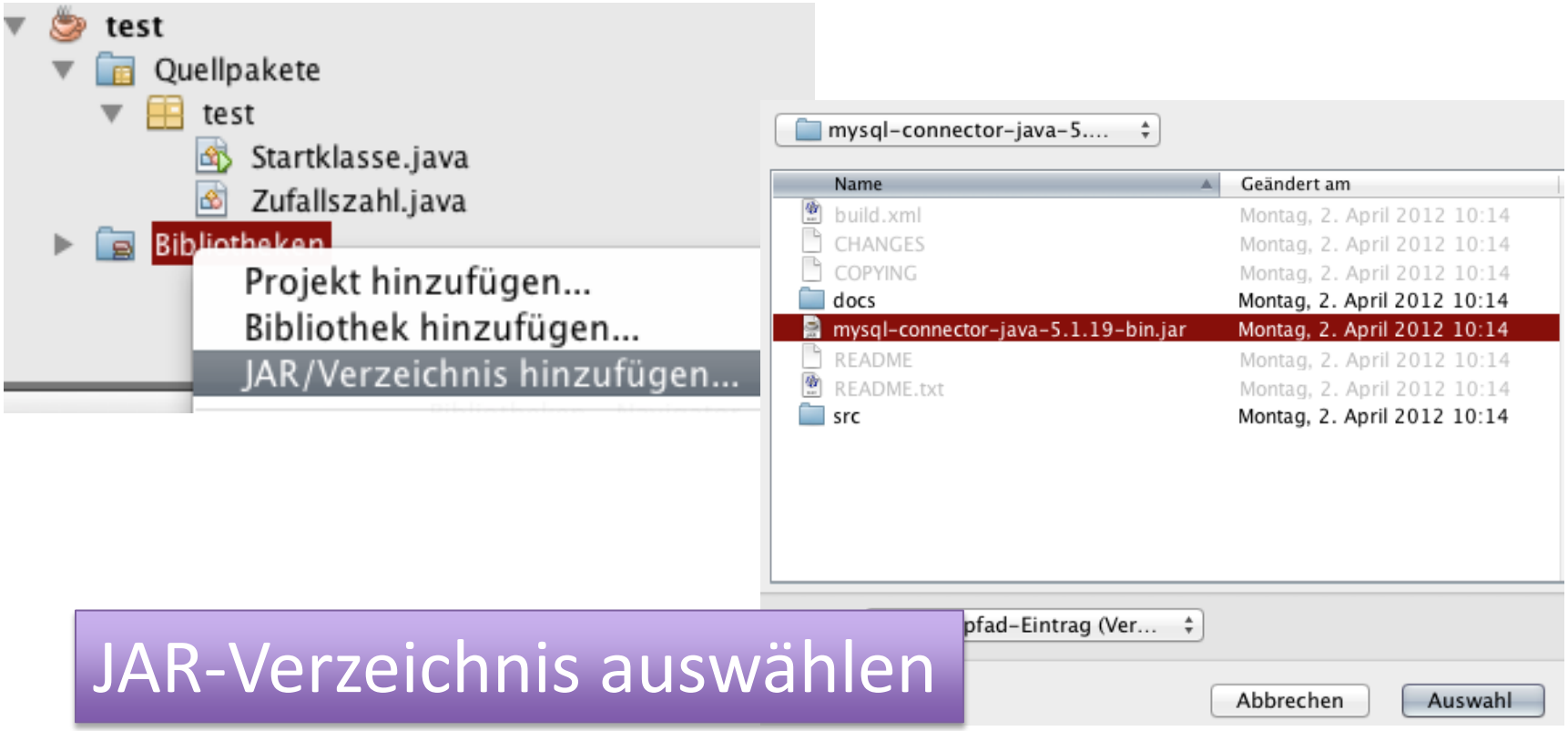
erhältlich hier:

www.mysql.de/downloads/connector/j/

(runterladen, entpacken)

Vorarbeiten

- JDBC-Treiber in die Bibliothek importieren:

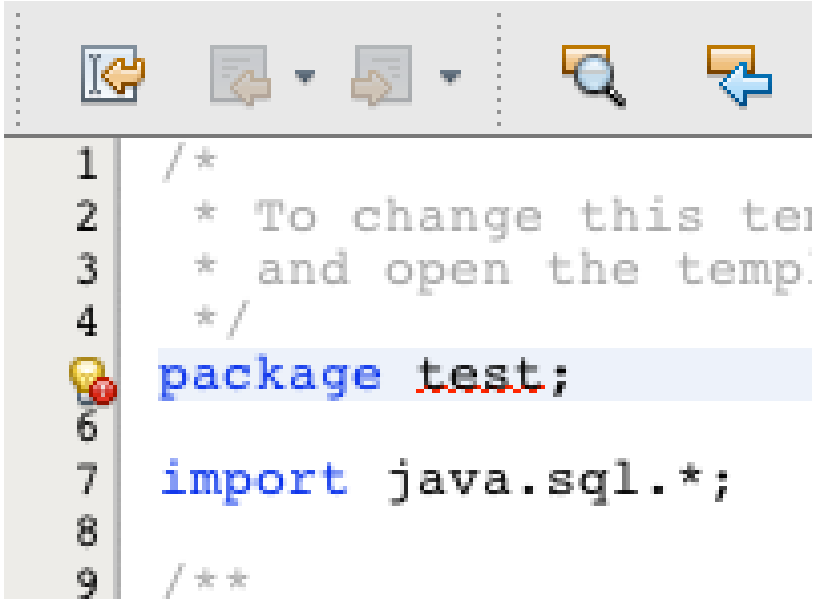


JAR-Verzeichnis auswählen

Vorarbeiten

- java.sql.* importieren

am besten von Hand; automatischer Import durch IDE (Netbeans, Eclipse) kann nerven.



Verbindung zur DB herstellen

KURZFASSUNG

```
private Connection verbindung;
// Objekt der Klasse Connection

verbindung = DriverManager.getConnection
                    (url, username, passwort);
// DB-Daten (Tabellenname, DB-Server ... für
  url)

// Zugangsdaten (Parameter für Methode
  getConnection)
```

Verbindung zur DB herstellen

Wir brauchen:

1) Objekt der Klasse Connection

als Attribut/Instanzvariable:

```
private Connection verbindung = null;
```

Verbindung zur DB herstellen

Wir brauchen:

2) Verbindungsdaten

als Attribute/Instanzvariablen oder Parameter an den Konstruktor:

```
private String dbName = "Datenbankname";
private String server = "localhost:3306";
private String user   = "root";
private String passwort = "";
```

3306 = Port für Windows
Mac/MAMP: 8889, PW: "root"

Verbindung zur DB herstellen

Wir brauchen:

3) JDBC-Verbindungsurl

Format:

jdbc:mysql://localhost:3306/artikel

Zusammenbauen mit:

```
String url =
  "jdbc:mysql://"+server+"/"+dbName;
```

Verbindung zur DB herstellen

Wir brauchen:

4) Java-Befehl, um Verbindung herzustellen

```
this.verbindung =
  DriverManager.getConnection(url, user, pass);
```

Achtung: Diese Zuweisung lässt sich NICHT bei der Deklaration der Attribute realisieren, weil sie mit einem try-catch-block umgeben werden muss!

Verbindung zur DB herstellen

Zusammenfassung:

```
private Connection verbindung = null;

private String dbName = "Datenbankname";
private String server = "localhost:3306";
private String user  = "root";
private String passwort = "";

String url =
   "jdbc:mysql://"+server+"/"+dbName;
// in einer Methode dann:
this.verbindung =
   DriverManager.getConnection(url, user, passwort);
```

Verbindung zur DB herstellen

verbindung = … muss
In einer Methode stehen!
(nicht bei den Attributen!)

Zusammenfassung:

```
private Connection verbindung = null;
this.verbindung =
   DriverManager.getConnection("jdbc:mysql://localhost:
   3306/Datenbankname", "root", "");
```

Kurzversion

```
// statt:
private String dbName = "Datenbankname";
private String server = "localhost:3306";
private String user   = "root";
private String passwort = "";

String url =
   "jdbc:mysql://"+server+"/"+dbName;
// in einer Methode dann
this.verbindung =
   DriverManager.getConnection(url, user, passwort);
```

Verbindung zur DB herstellen

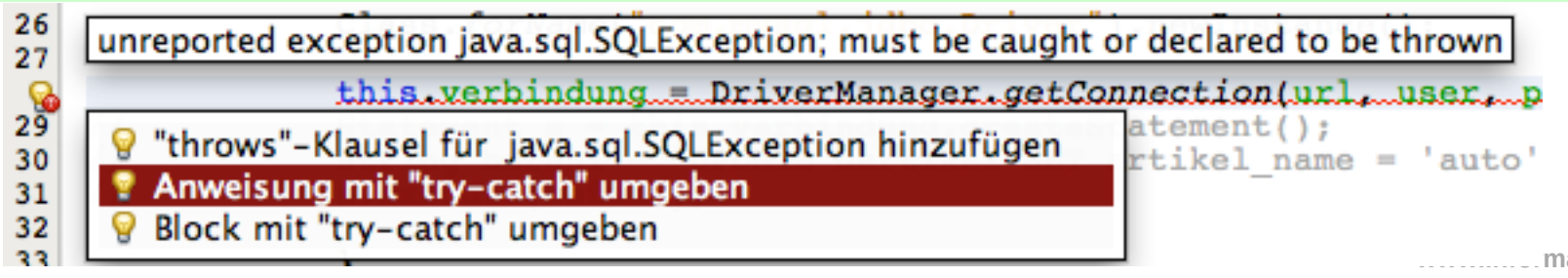
Achtung: Exceptions abfangen

oben aus Platzgründen nicht realisiert

```
try {
    String url = "jdbc:mysql://" + dbserver + "/" + d
    System.out.println(url);
    this.verbindung = DriverManager.getConnection(url
    this.stmt = this.verbindung.createStatement();

}
catch (Exception e) {
    System.out.println(e);
}
```

Mit IDE-Hilfe: Linke Maustaste auf Fehler-Markierung, "Anweisung mit try-catch umgeben":



Verbindung zur DB herstellen

Verbindung gesamt:

```
public class VerbindungHerstellen
{
    private Connection verbindung = null

private String dbName = "test";
private String server = "localhost:8889"
private String user = "root";
private String passwort = "root";
String url =
    "jdbc:mysql://"+server+"/"+dbName;

public VerbindungHerstellen() {
    try{
        this.verbindung =
DriverManager.getConnection(url, user, p
    }
    catch(Exception e) {
        System.out.println(e);
    }
```

1: Objekt Connection

2: Verbindungsdaten

3: Verbindungs-URL

4: Java-Befehl (DriverManager)

Abfragen ausführen

Wir brauchen:

1) ein Objekt der Klasse Statement (beinhaltet den MySQL-Code)
2) ein Objekt der Klasse ResultSet (speichert bei SELECT-Abfragen die Ergebnisse)

```
private Statement stmt = null;
private ResultSet ergebnismenge
            = null;
 // Resultselt nur bei SELECT-
Abfrage nötig
```

Abfragen ausführen

Wir brauchen:

1) ein Objekt der Klasse Statement (für Ausfühung des MySQL-Codes)
2) ein Objekt der Klasse ResultSet (speichert bei SELECT-Abfragen die Ergebnisse)

```
private Statement stmt = null;
this.stmt = verbindung.createStatement();

private ResultSet rs = null;
   // nur bei SELECT-Abfrage nötig
```

Abfragen ausführen

Abfragen zur Manipulation der Tabelle:

CREATE TABLE, UPDATE, INSERT INTO

```
private Statement abfrage = verbindung.createStatement();
private ResultSet rs = null;
```

```
String sqlBefehl =
  "UPDATE bla SET name='urgh';";
this.abfrage.executeUpdate(sqlBefehl);
```

- *executeUpdate() für insert, update, delete;*
- *executeQuery() für select*
- *execute() → gibt TRUE zurück, wenn Ergebnis = ResultSet, ansonsten FALSE*

Abfragen ausführen

SELECT-Abfragen ... ausführen

```
private Statement abfrage = verbindung.createStatement();
private ResultSet rs = null;
```

```
String sqlBefehl =
   "SELECT * FROM artikel";
this.rs = this.abfrage.executeQuery(sqlBefehl);
```

→ Ergebnis der Abfrage wird als "ResultSet" (this.rs) gespeichert.

Abfragen ausführen

SELECT-Abfragen ... Ergebnisse darstellen

```
String sqlBefehl =
   "SELECT * FROM artikel";
this.rs = this.abfrage.executeQuery(sqlBefehl);

while (this.rs.next()){
   // so lange rs.next() true ist, gibt es noch
   // weitere Datensätze anzuzeigen
       int id = this.rs.getInt("id");
       String name = this.rs.getString("name");
       System.out.println("ID: " + id + ", Name:" + name);
    }
```

"id" und "name" = Feldnamen der befragten DB-Tabelle

Nützliche Methoden

```
String sqlBefehl = "SELECT * FROM artikel";
this.rs = this.abfrage.executeQuery(sqlBefehl);

this.rs.beforeFirst();
   // Satzzeiger vor die erste Zeile setzen
this.rs.last();
   // Springt zum letzten Datensatz
this.rs.absolute(int);
   // Springt zum angegebenen Datensatz. Bei negativem Wert
   wird vom letzten Datensatz rückwärts gezählt.
this.rs.relative(int);
   // Springt vom aktuellen Datensatz aus um die angegebene
   Zahl an Datensätzen weiter. Achtung: Es MUSS einen
   aktuellen Datensatz geben; nach beforeFirst() gibt es
   keinen ausgewählten Datensatz!
```

DB-Verbindung schließen

```
verbindung    =
   DriverManager.getConnection(url, user, pass);
Statement s    =
   v.getVerbindung().createStatement();

   // DB-Handling ....
   // ...

verbindung.close();
s.close();
// Beim Schließen des Statement-Objekts wird das
   ResultSet ebenfalls geschlossen
```

Lesenswert dazu:
http://blog.shinetech.com/2007/08/04/how-to-close-jdbc-resources-properly-every-time/

Aufgabe

1) Versuchen Sie mittels Java eine KLEINE Tabelle in Ihrer DB zu erstellen (3 Attribute). Fügen Sie (ebenfalls über Java) ein paar Datensätze ein.
2) Führen Sie einige SELECT-Abfragen aus. Die Namen der Tabellenspalten kennen Sie ja von Aufgabe 1.

TIPP:

Schreiben Sie Ihr Programm gleich so, dass Sie die Klasse(n) bzw. Methoden später wieder verwenden können, z.B.

- `dbVerbindungAufbauen(user:String, pass:String, dbName:String, dbServer:String)`
- `dbManipulieren(sqlBefehl:String)`
- `dbSelectAbfrageAusfuehren(sqlBefehl:String)`
- `dbSchliessen()`