Java und Datenbanken: JDBC

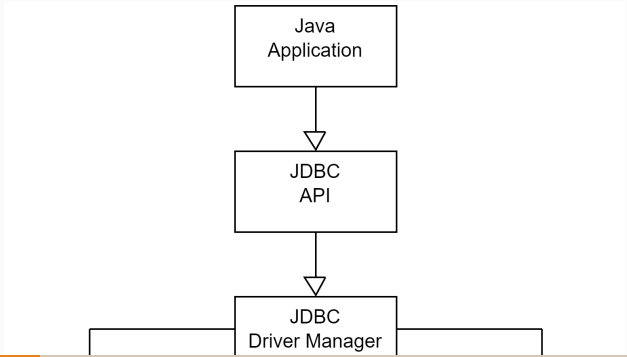
Carsten Gips (HSBI)

Motivation

- Mit Datenbanken interagieren, Daten senden und abfragen

JDBC

- **J**ava **D**ata**b**ase **C**onnectivity (JDBC) ist eine Java-API, um auf Datenbanken zuzugreifen
- Damit können Verbindungen zu Datenbank hergestellt und SQL-Statements ausgeführt werden.
- JDBC konvertiert die SQL-Datentypen in Java-Datentypen und umgedreht.
- Die JDBC API ist universal und Datenbanksystem unabhängig
- Die einzelnen Datenbanksystem-Hersteller stellen JDBC-Treiber zur Verfügung.
- Was machen die Treiber? Implementieren die von JDBC vorgegebene Schnittstelle,
damit der Treiber vom JDBC-Driver-Manager genutzt werden kann.
- Der JDBC Driver Manager lädt den Datenbanksystem spezifischen Treiber in die Anwendung.



Treiber Registrieren

Für unterschiedliche Datenbanksysteme gibt es unterschiedliche Treiber. Diese müssen in der Java-Anwendung registriert werden, um mithilfe von JDBC eine Verbindung zur Datenbank aufzubauen und Anweisungen zu verschicken.

Möglichkeit 1: Dynamsch zur Laufzeit `Class.forName()`

```
Class.forName("{datenbanktreiber}")
```

Beispiel:

Oracle:

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

MySQL:

```
Class.forName("com.mysql.jdbc.Driver");
```

Treiber-Registrationen kann so konfigurierbar und portierbar gemacht werden. (Man muss "nur" den String

Verbindung aufbauen

Mit drei Parametern.

```
String URL="jdbc:URL/TO/DATABASE";
String USER= "USER";
String PASSWORD = "PASSWORD"
Connection connection = DriverManager.getConnection(URL,USER,PASSWORD)
```

Mit einem Paramter. Username und Passwort werden in der URL angegeben.

```
String URL="jdbc:USER/PASSWORD/URL/TO/DATABASE";
Connection connection = DriverManager.getConnection(URL)
```

Mit Properties um Username und Passwort anzugeben.

```
java   String URL="jdbc:URL/TO/DATABASE";   Properties login = new Properties();   lo
```

Am Ende muss die Verbindung zur Datenbank geschlossen werden.

```
connection.close();
```

Statements

- Mit `Statement`s werden SQL-Befehle erstellt, die dann an die Datenbank gesendet werden können.

Statement erstellen mithilfe des `Connection`-Objekts

```
Statement st= connection.createStatement();
```

`ResultSet`

- Alle SQL-Statements die Daten aus der Datenbank lesen, geben diese als `ResultSet` zurück und kann sich wie eine Tablel vorgestellt werden.
- Das `ResultSet`-Objekt hält dann einen Pointer auf die aktuell betrachtete Reihe in der Tabelle.
- `ResultSet`s können auch Konfiguriert werden
  - Zugriffsrechte (`RSConcurrency`)
    - `CONCUR_READ_ONLY` (default): Nur Lesezugriff auf die Daten.
    - `CONCUR_UPDATABLE`: Daten können über das `ResultSet` geupdated werden.
  - Scrollbarkeit (`RSType`)
    - `TYPE_FORWARD_ONLY` (default) Pointer kann nur Vorwärts bewegt werden
    - `TYPE_SCROLL_INSENSITIVE`: Pointer kann Vorwärts und Rückwärts bewegt werden
    - `TYPE_SCROLL_SENSITIVE`: Pointer kann Vortwärts und Rückwärts bewegt werden, zeitgleich werden Änderungen in der Datenbank berücksichtigt (das `ResultSet` updated sich)
- Um das `ResultSet` zu konfigurieren, müssen die Parameter im `Statement` gesetzt werden `Statement st= connection.createStatment(RSType,RSConcurrency)`.

Beispiel Abfragen

Datensätze aus der Datenbank abfragen:

```
String sql= "SELECT * FROM USER";
ResultSet rs = st.executeQuery(sql);

while(rs.next){
    System.out.println("ID:" + rs.getInt("id"));
    System.out.println("Username:" + rs.getString("name"));
    System.out.println("Age:" + rs.getInt("age"));
}
rs.close();
```

Datensätze in der Datenbank hinzufügen:

```
String sql="INSERT INTO User VALUES ('Wuppi Fluppi',22)";
st.executeUpdate(sql);
sql="INSERT INTO User VALUES ('Tutti Frutti ',100)";
st.executeUpdate(sql);
```

SQL-Exceptions

- Auch mit JDBC kann es zu Fehlern/Probleme kommen.
  - Fehlerhafte Statements
  - Verbindungsprobleme
  - Fehler in den Treibern oder der Datenbank selber
- Daher ist Exceptionhandling besonders wichtig.
-

```
try {
    // do something
}
catch (Excpetion e){
    //ups
    e.printStackTrace();
}
finally{
    connection.close();
}
```

Wrap-Up

- JDBC ist eine API um mit Datenbanken zu interagieren
- JDBC verwendet einen Driver-Manager
- gibt unterschiedliche treiber
- how to connection aufbauen
- how to statement senden
- how to result auswerten ...

LICENSE

Exceptions

- TODO (what, where, license)