Java Database Connectivity

Forschungsprojekt

Inhaltsverzeichnis

Einleitung

Für Java gibt es diverse Plugins mit Attributen aus verschiedenen Programmierbereichen, die die Arbeit mit dieser Hochsprache immer wieder interessant gestalten, ständig neue Entwicklungsmöglichkeiten aufzeigen und Java damit als gutes Forschungsprojekt kennzeichnen.
Die JDBC API ist so ein Plugin - ausgestattet mit der Fähigkeit, eine stabile Verbindung zu einem Host bzw. Datenbank aufzubauen, SQL-Anweisungen zu formulieren, auszuführen und mögliche Resultate darzustellen, oder zu modifizieren.

Verschiedene Schritte sind notwendig, eine Verbindung aufzubauen und sinnvolle SQL-Anweisung umzusetzen. In diesen Beleg werde ich sowohl auf die grundlegenden Attribute bzw. Schritte eingehen, die es bedarf, die JDBC API zum Laufen zu bringen, als auch das von mir erarbeitet Programm in seinen Fähigkeiten vorzustellen und als gutes Forschungsprojekt zu präsentieren.

1 JDBC API

1.1 Driver

Es gibt diverse Driver, die die Kommunikation mit unterschiedlichen Datenbank-Servern, etwa einer MYSQL-DB oder Oracle-DB regeln. Meist wird beim ersten Aufruf der JDBC API der DriverManager benötigt, der aus einer Liste diverser Driver die passende Klasse wählt.

```
try{
 DriverManager.registerDriver(new com.mysql.jdbc.Driver());
}
catch(SQLException ex){...}
```

Wurde diese Prozedur durchlaufen, wird der Aufruf des Manager nicht mehr benötigt und die entsprechenden Attribute über die Class aufgerufen.

```
try{
 Class.forName("com.mysql.jdbc.Driver");
}
catch(ClassNotFoundException ex){...}
catch(IllegalAccessException ex){...}
catch(InstantiationException ex){...}
```

1.2 Connection

Für die Verbindung selbst, ist es wichtig neben der Driverklasse noch die URL zum Host zu formulieren. Dieser Schritt wird wohl in allen Programmierbereichen rund um Datenbanken durchlaufen, z.B. bei der Verbindung eines FTP-Programms mit einem Host. Vergleichbar mit diesen Bereichen braucht es URL zum Host bzw. Datenbanknamen, Benutzernamen und Passwort. Programmiertechnisch lassen sich diese Eigenschaften unterschiedlich in der URL für die Verbindung umsetzen. Dabei ist darauf zu achten, das es diverse Driver mit speziellen Aufgaben gibt und sich so die URL verschieden gestaltet.

```
try{
 Connection connection=DriverManager.getConnection(url);
}
catch(SQLException ex){...}
```

1.2.1 MYSQL

Einfachste URL ist wohl die der Verbindung über MYSQL. Beispiel einer URL, die gleich Benutzername und Passwort enthält.

```
jdbc:mysql://localhost/Mitarbeiter?user=admin&password=007
```

Jeder der mit MYSQL und PHP schon mal gearbeitet hat, wird die Grundzüge sofort wieder erkennen.

```
jdbc : mysql : // DB / Tabelle ? Benutzernamen = Wert & Passwort = Wert
```

1.2.2 Oracle

Die URL zu einer Oracledatenbank ist vielschichtiger und es werden mehr Eigenschaften ausformuliert. Zum einem muss die Art des Drivers genannt werden, beispielsweise :thin: oder :oci8: und zum anderen kommen Werte wie Port, bzw. SID zum tragen.

```
jdbc:oracle:thin:admin/007@Mitarbeiter:1521:orcl
jdbc:oracle:oci8:admin/007@Mitarbeiter:1521:PO8
```

```
jdbc : oracle : Drivertyp : Benutzernamen / Passwort @ DB : Port : SID
```

In Form und Struktur können sich die Möglichkeiten der URL sowohl von den verschieden Datenbank-Servern, als auch von der Unterbringung der Parameter, wie bereits erwähnt, unterscheiden.

3 Beispiele der Unterbringung der Parameter:

```
DriverManager.getConnection(url)

oder
DriverManager.getConnection(url,user,pass)

oder
DriverManager.getConnection(url,new Properties())
```

1.3 SQL-Anweisungen

Dieser Programmierbereich ähnelt stark dem SQL + PHP Standard. Simple SQL-Anweisungen in Javacode eingebettet.

Beispiel MYSQL + PHP

```
$con=mysql_connect(host,user,pass);
$sql="CREATE TABLE table_name(
 ID INT NOT NULL AUTO_INCREMENT,
 value0 VARCHAR(255) NOT NULL,
 ...
 PRIMARY KEY(ID))";
mysql_select_db('db_name');
$result=mysql_query($sql,$con);
```

Beispiel SQL + Java

```
Connection connection=DriverManager.getConnection(url,user,pass);
String sql="CREATE TABLE table_name(
 ID INT NOT NULL AUTO_INCREMENT,
 value0 VARCHAR(255) NOT NULL,
 ...
 PRIMARY KEY(ID))";
connection.createStatement().executeUpdate(sql);
```

Es gibt kaum nennenswerte Unterschiede, außer in Form der Differenzierung zwischen den verschiedenen Arten der SQL-Anweisung. Bei SQL + PHP haben die Anweisungen ungefähr gleiche Form. Create ist aus den letzten Beispiel bekannt, hier noch kurz Select und Insert, um die gleiche Form der Befehle zu betonen.

```
$sql = "INSERT INTO table_name (...) VALUES (...)";
mysql_select_db('db_name');
$result = mysql_query($sql,$con);

$sql="SELECT columns FROM table_name";
mysql_select_db('db_name');
$result = mysql_query($sql,$con);
```

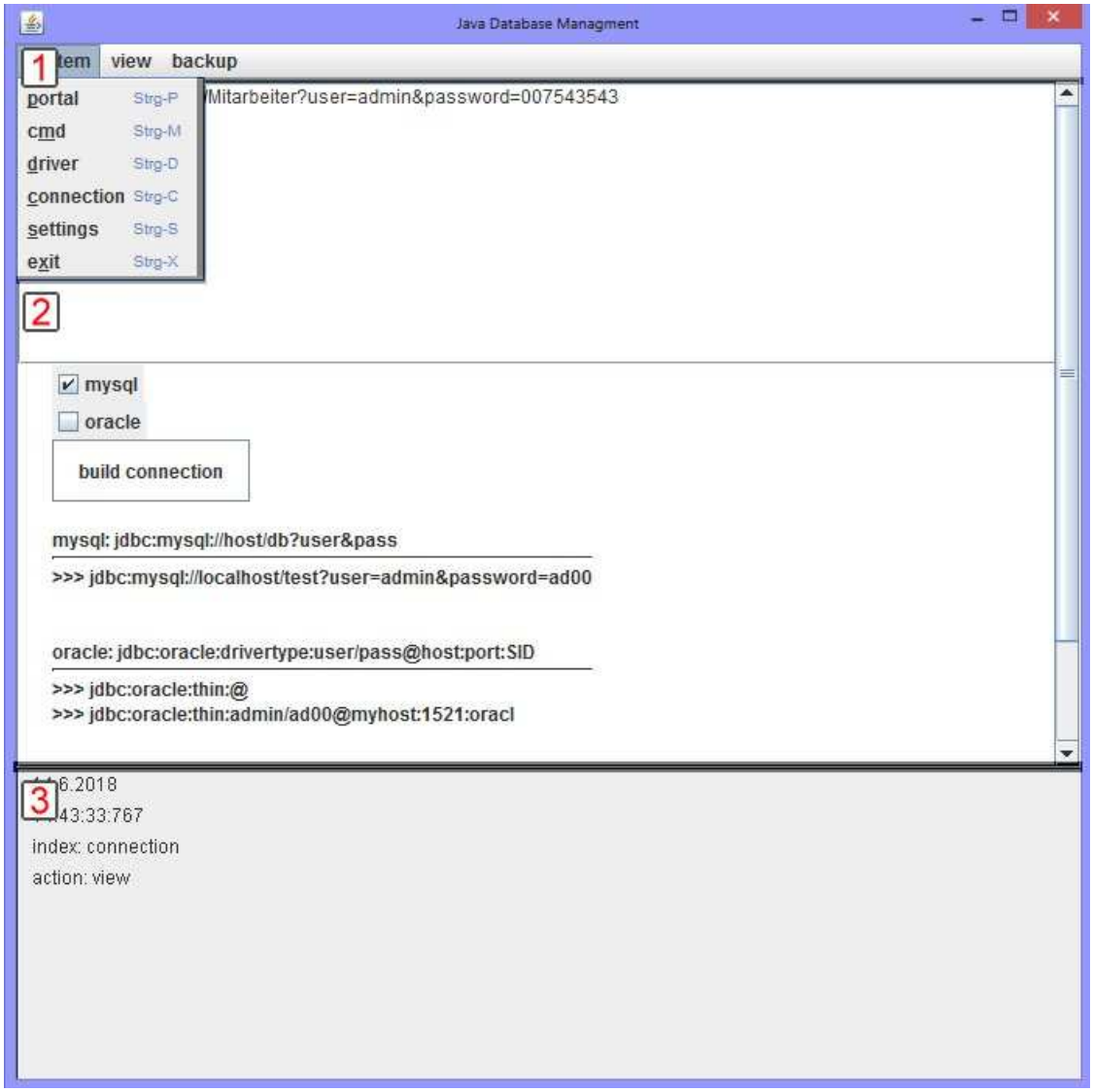
In Java wird hier zwischen den einzelnen Anweisungen mehr differenziert. Naben dem Bereich der Statements gibt es auch noch die der PrepareStatements, zur Arbeit mit eingebetteten Variablen, d.h. auch wenn sich Programmierung mit JDBC gerade für die SQL-Anweisung sehr einfach hält, wurde hier mehr ausformuliert als beispielsweise bei MYSQl + PHP.

```
connection.createStatement().executeUpdate(
 "INSERT INTO  table_name (...) VALUES (...)"
);
connection.createStatement().executeQuery(
 "SELECT columns FROM table_name"
);
```

2 Programm

Das Programm besteht aus 3 Modulen, die zum einen eine einfache Verbindung zu einer Datenbank aufbauen, die Möglichkeit geben, entsprechende Tabelle zu erstellen bzw. zu warten, als auch die unterschiedlichen Attribute erklären soll.
Einfach ausgedrückt, kann man bei der Handhabung des Programmes gleich einen tieferen Einblick in die Arbeit mit der JDBC API bekommen. Das habe ich umgesetzt, indem ich die einzelnen Arbeitsschritte bei der Einrichtung der Verbindung und die daraus resultierende Bearbeitung der Daten in Programmmodule eingeteilt habe. Bestes Beispiel dafür ist das Modul system, wo für jeden einzelnen Schritt ein separates Untermodul aufgerufen wird. Dabei kann man 'step by step' lernen, wie JDBC mit den einzelnen Eigenschaften arbeitet.

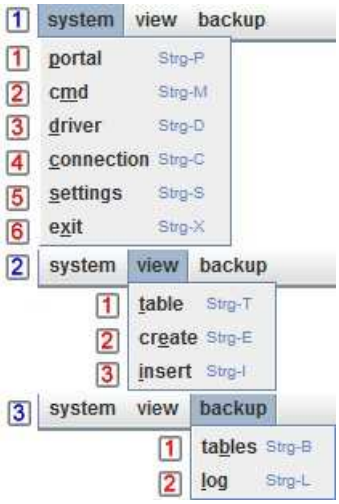
2.1 Aufbau der Benutzeroberfläche



1. Menu mit Hotkey's
2. Index-Panel für Eingaben und Auswahlbereiche
3. Log-Panel für die Darstellung von Index-Wechsel bzw. Aktion

- in Englisch und Deutsch

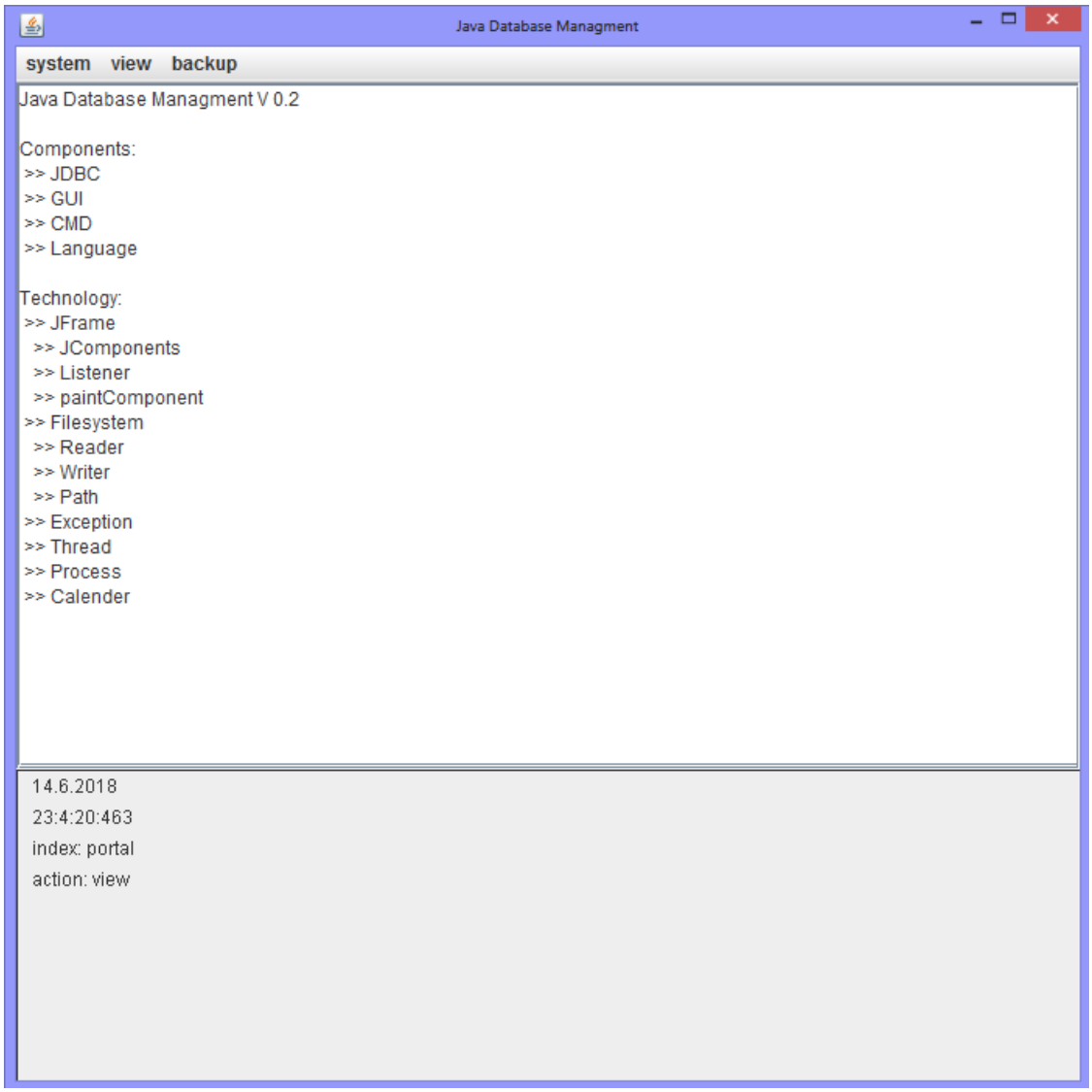
2.2 Menu



1. Systembereich mit Zugang zum CMD und dem Verbindungseinstellungen
    1. Portal
    2. CMD
    3. Treiber
    4. Verbindung
    5. Gesamteinstellungen
    6. Programm schließen
2. View für SQL-Aktionen
    1. Tabelle
    2. Create
    3. Insert
3. Backup listet Tabellenbackups und Logs auf
    1. Tabellen
    2. Logs

2.3 system

- cmd: manuelles Einrichten des Servers und Verknüpfung der jar's
- driver: Auflistung zweier möglicher Driverklassen mit den Elementen
- connection: Ausformulierung URL und Beispiele
- settings: Kurzfassung vorheriger Schritte (unvollständig)



2.3.1 cmd

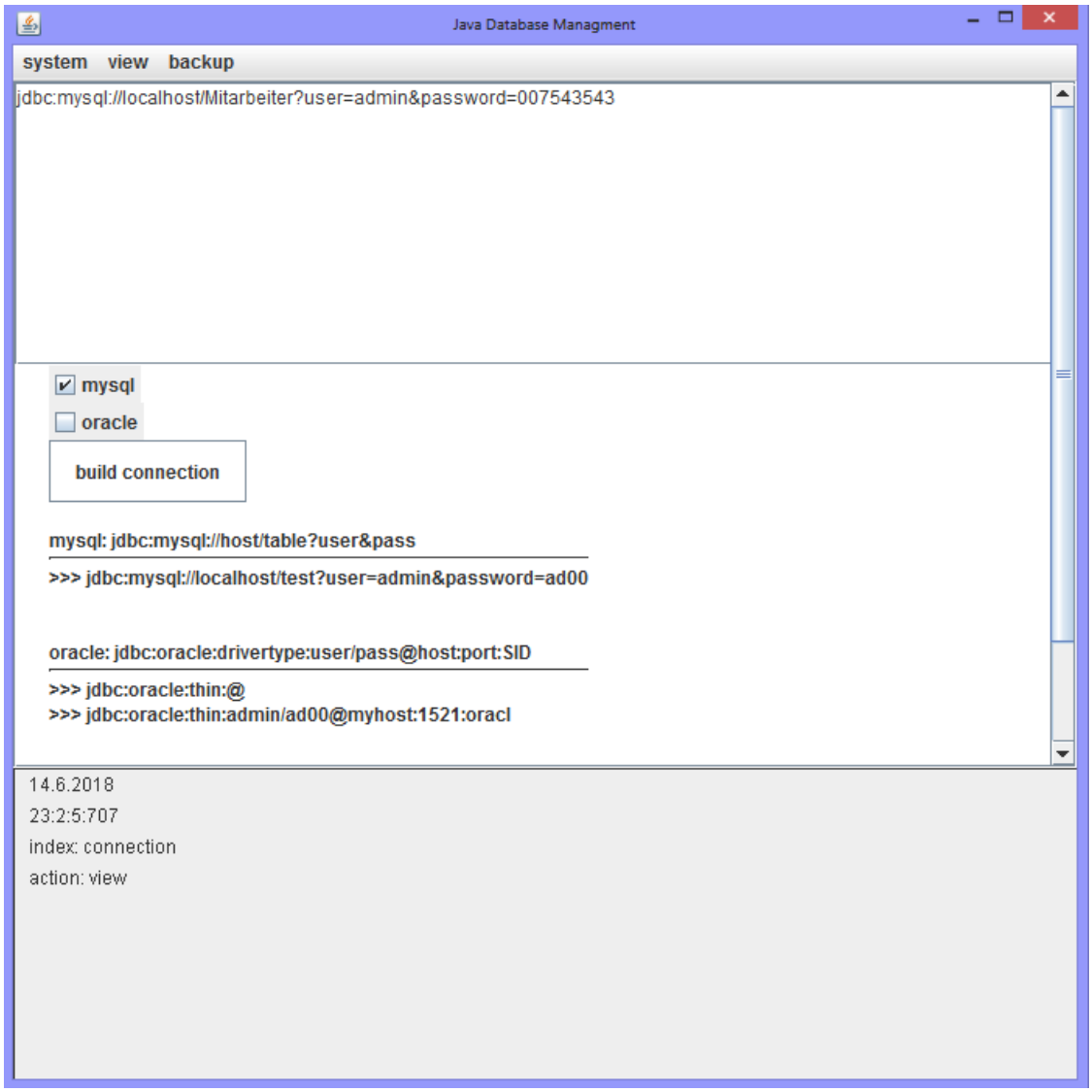
Es ist nicht immer möglich, mit einer Buildumgebung zu arbeiten, d.h. für die JDBC API sind jar's notwendig, die erst entsprechend eingebunden werden müssen. Dafür sind Buildtools, wie etwa Maven, oder IDE's wie Eclipse vorteilhaft, aber nicht immer zur Hand. Dieses Untermodul kompensiert diese Schwäche und stellt einen Zugang zum Commandopanel dar. Auch die Arbeit mit dem Server wird erleichtert, da kein separates cmd-Feld geöffnet werden muss und mögliche Datenbankeinstellungen über das Programm vorgenommen werden können. Es handelt sich dabei um kein Pflichtfeld, sondern eine sehr nützliche Erweiterung.

2.3.2 driver

Als Leihe steht man vor den Fragen, was sind Driver, welche gibt es und wie werden sie eingebunden. Bei diesen Fragen hilft dieser Menupunkt, da die wichtigen Fragen für 2 der entscheidenden Driver erläuternd aufgelistet werden. Darüber hinaus wird der Status der Driver ermittelt, d.h. in welcher Form ist der Driver vorhanden. Muss er erst mittels DriverManager bzw. dessen newInstance initialisiert werden, oder kann man schon auf die Klasse zurückgreifen, da bereits alles geladen ist. Sehr hilfreich, wenn man sich neu mit der Thematik beschäftigt.

2.3.3 connection

Die wichtigen Punkte der Verbindung werden hier beleuchtet und stehen zur Auswahl. Es besteht die Möglichkeit der Wahl der Driver, die im vorherigen Menupunkt aufgelistet wurden und der Ausformulierung der URL. Ich bin von der Idee abgewichen, die Parameter der Verbindung einzeln aufzulisten und habe mich dafür entschieden nur mit de URL zu arbeiten. Den Benutzernamen und das Passwort separat abzufragen, widersprach dem Prinzip der Einzelabfrage durch die URL und dem letzteren konnte ich mehr abgewinnen. Kleine Beispiele gehen auf die Struktur der URL ein, die ich im Kapitel 1.2 erläutert habe.



2.3.4 settings

Der erfahrende Benutzer soll die Möglichkeit haben, die gesamten Einstellungen der vorherigen Menupunkte in einem Arbeitsschritt durchführen zu können. Da die Parameter für die Verbindung gemeinsam in einer Klasse gespeichert werden, ist deren Auflistung nicht sonderlich schwer.

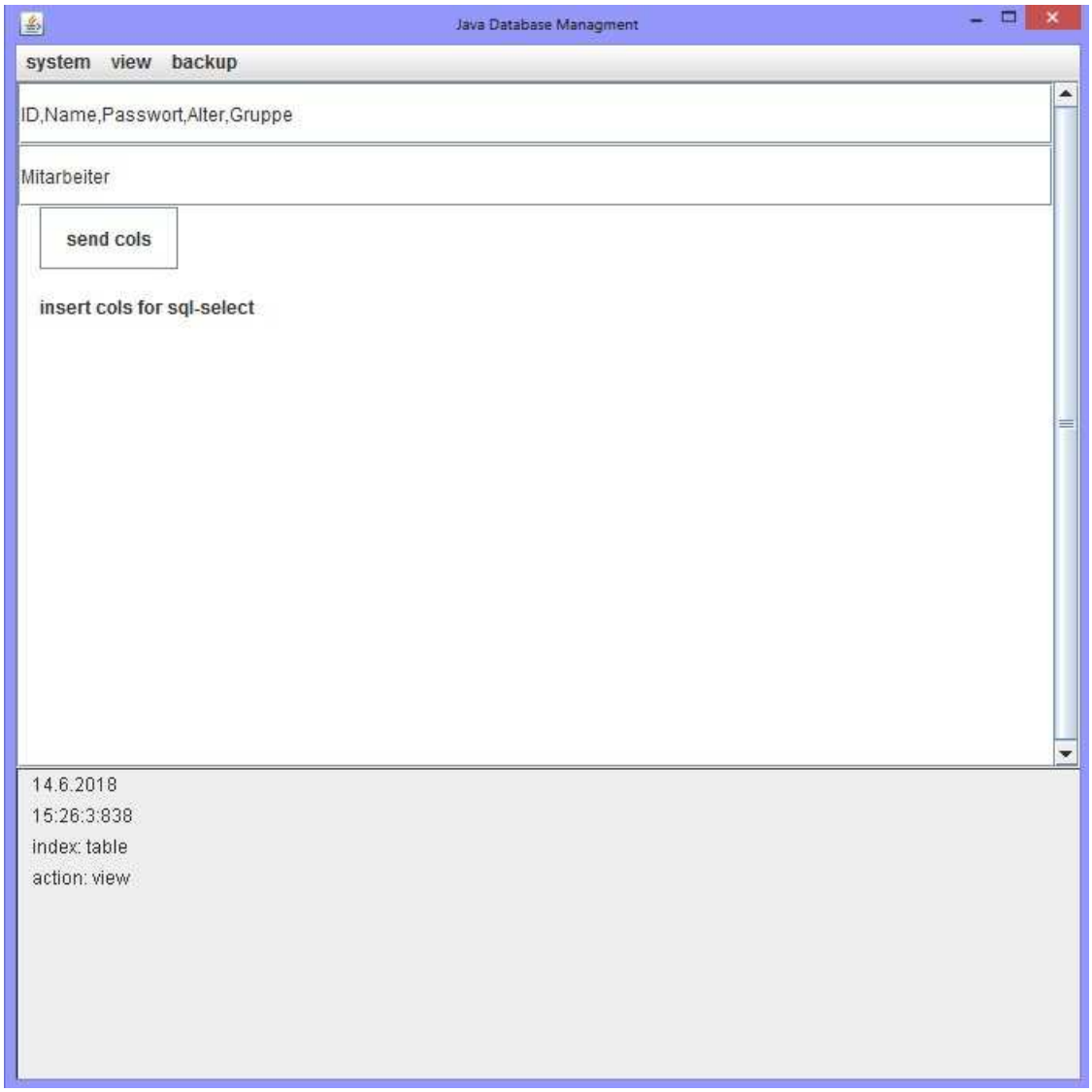
2.4 view

- view table: komplette Datensatz der ausgewählten Tabelle mit Updatefunktion
- create: Möglichkeit Tabelle zu erstellen
- insert: Einfügen einer neuer Zeile

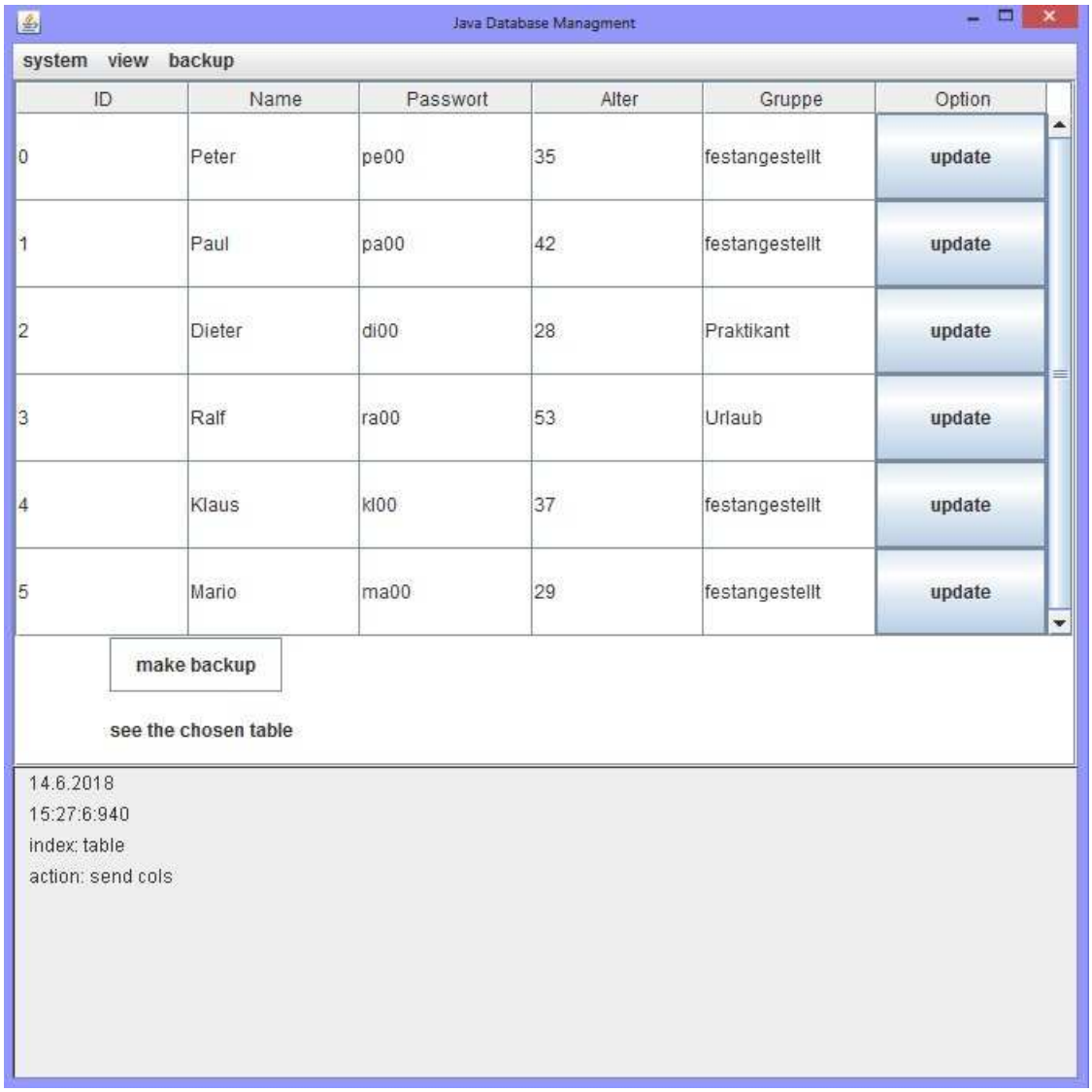
2.4.1 view table

Mittels Select wird ein kompletter Datensatz einer ausgewählten Tabelle abgebildet. Dabei wird zwischen Spaltennamen und Zeileninhalt differenziert bzw. bei der Auflistung jeder Zeile eine Updatefunktion eingebettet, die über den Index die entsprechende Position im Datensatz ermittelt und mit dessen Hilfe der Inhalt aktualisiert werden kann. D.h. es verstecken sich hier zwei SQL-Anweisungen: Select und Update.

Spalten.- und Tabellenname für SELECT



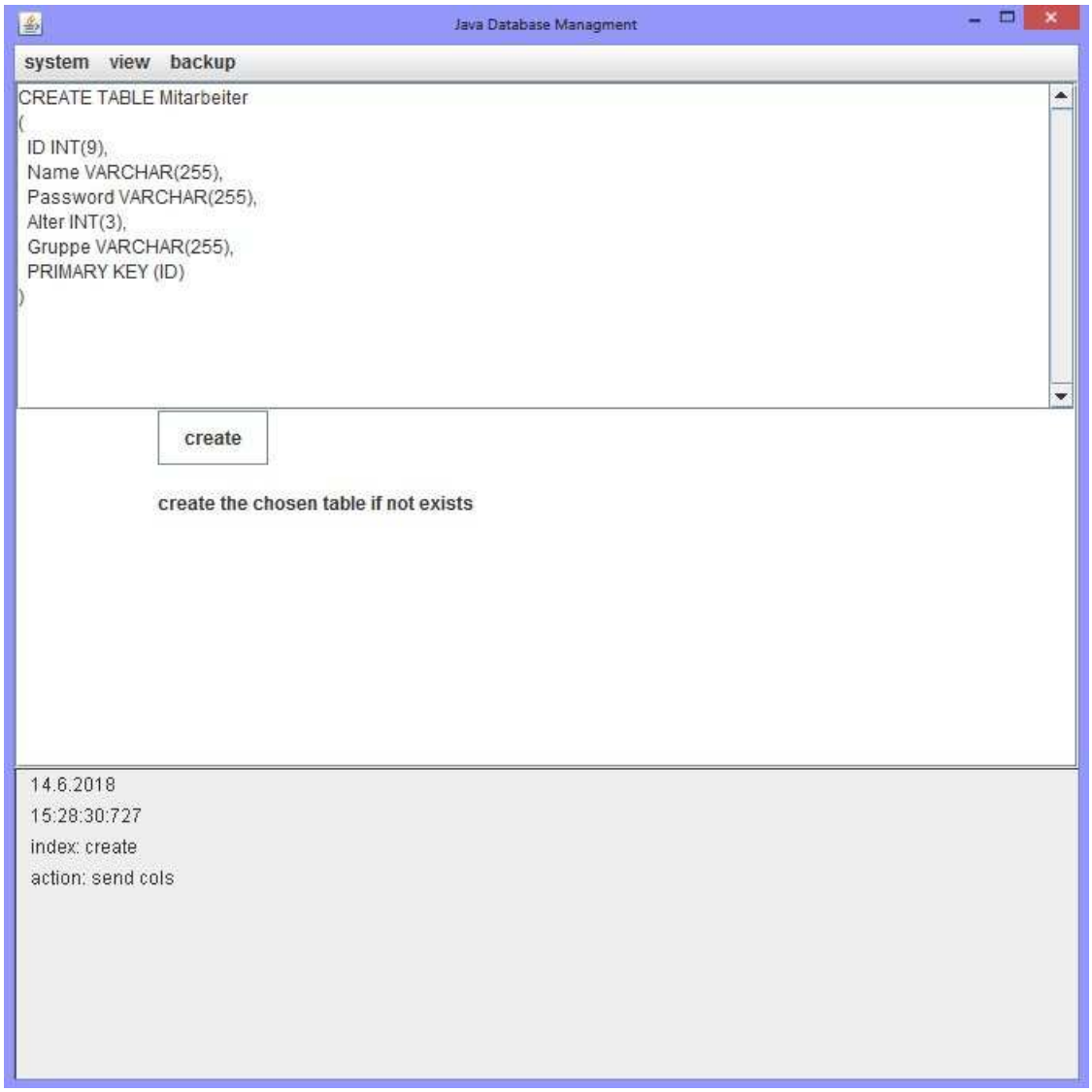
View mit Datensatz



Updatefunktion

2.4.2 create

Über ein Inputfeld (JTextArea) kann eine neue Tabelle erstellt werden, bzw. wenn die Standardtabelle 'Mitarbeiter' nicht existiert, kann sie über diesen Programmbereich initialisieren.



2.4.3 insert

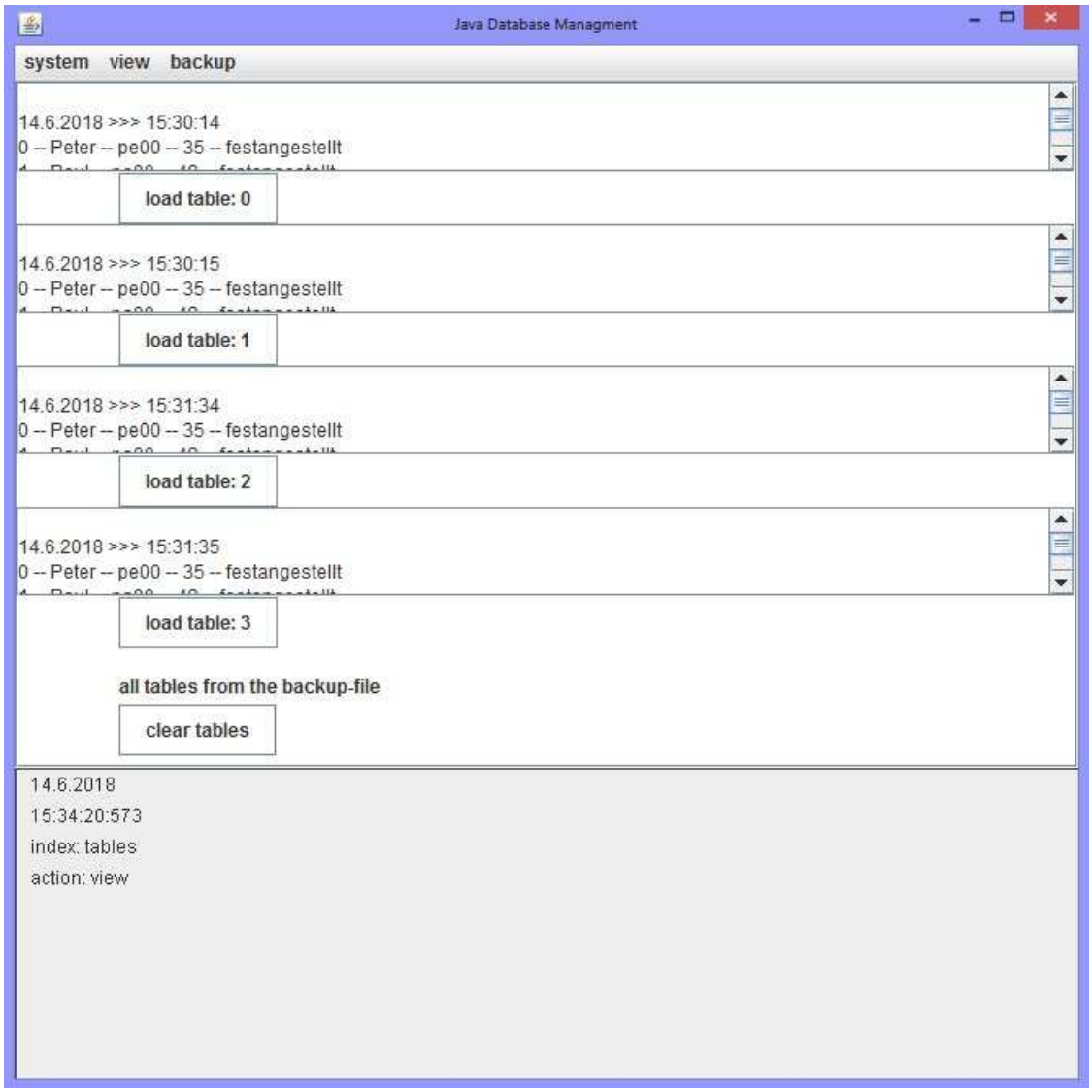
Ähnlich der Createfunktion wird über ein einzelnes Feld eine neue Zeile in den Datensatz geschrieben. Leider wird noch nicht zwischen einzelnen Datentypen unterschieden. Bei der nächsten Überarbeitung bzw. bis zur Projektabgabe wird die Differenzierung zwischen Datentypen eingebunden sein.

2.5 backup

- backup: Tabellenbackups werden aufgelistet (unvollständig)
- log: alle Programmbewegungen werden aufgelistet

2.5.1 tables

Vergleichbar mit der Logfunktion, kann hier eine Liste von Tabellenbackups ausgegeben werden. Auch sehr hilfreich, bei mehreren Projektteilnehmern bzw. als Nothalt bei fehlerhaften Änderungen.



2.5.2 log

Jede einzelnen Bewegung innerhalb des Programmes wird gespeichert und kann über diese Funktion als Gesamtes aufgelistet werden, d.h. wenn man sich über das Menu zu den verschiedenen Programmmodulen und Untermodulen bewegt, wird das im logpanel angezeigt und in einer Datei mittels Filesystem gespeichert. Diese Funktion kann sehr hilfreich sein, wenn man Beispielsweise mit mehreren Leuten an einem Projekt arbeitet und ein lückenloses Protokoll benötigt.

2.6 logpanel

Das Logpanel ist der untere Bereich des Programms, der Aufschluss über Status und Programmbewegung gibt. Arbeitet im Millisekundentakt und listet Programmmodul und aktuelle Aktion auf.

3 verwendete Technologie und Komponenten

Diese Programm besteht aus sehr vielen einzelnen Elementen, die die Arbeit mit der JDBC API stark vereinfachen und das Ganze ansehnlich machen. Hier ein paar Beispiele:

- cmd: macht Buildumgebung unnötig, erleichtert Serverkommunikation
- filesystem: lückenlose Protokoll von Programmbewegung bzw. Aktionen und Tabellenbackups
- thread: Echtzeitdarstellung von Programmbewegung und Aktionen
- language: Sprachauswahl

Die einzelnen Attribute wurden für Java typisch gekapselt, sodass jede Eigenschaft entsprechende Klasse bzw. Funktionen aufweist. In der Programmcodestruktur lassen sich alle Komponenten isoliert betrachten, was sowohl das Verständnis, als auch die Entwicklung bzw. Wartung stark vereinfacht.

3.1 jdbc api

Die wichtigsten Komponenten sind:

- driver
- connection
- statement
- prepareStatement

Alle grundlegenden Elemente sind umgesetzt und für den Benutzer vereinfacht dargestellt. Wichtiges Augenmerk ist die Ausformulierung einzelner Prinzipien, die über den einfachen Standardgebrauch hinausreicht.

Beispiel checkDriverStatus():

```
public boolean checkDriverStatus(boolean check,String driverKey){
 int count=0;
 for(Enumeration en=DriverManager.getDrivers();en.hasMoreElements();count++){
  if(driverKey.equals(en.nextElement().getClass().getName())){check=true;}
 }
 return check;
}
```

Beispiel driverRegisteredAs():

```
public String driverRegisteredAs(boolean check,String driverKey,String loaded){
 try{Class.forName(driverKey);check=true;loaded="as Class";}
 catch(ClassNotFoundException ex){check=false;}
 if(check==false){
  try{Class.forName(driverKey).newInstance();check=true;loaded="as new instance\nfrom Class";}
  ...
 }
 if(check==false){
  if(driverKey.equals("com.mysql.jdbc.Driver")){
   try{DriverManager.registerDriver(new com.mysql.jdbc.Driver());check=true;...}
   catch(SQLException ex){check=false;}
  }
  if(driverKey.equals("oracle.jdbc.driver.OracleDriver")){
   try{DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());check=true;...}
   catch(SQLException ex){check=false;}
  }
 }
 return loaded;
}
```

3.2 GUI und deren Komponenten

Die wichtigsten Komponenten sind:

- JFrame
- getContentPane
- JOptionPane
- JPanel
- JMenuBar
- JMenu
- JMenuItem

- JLabel
- JTextField
- JTextArea
- JButton
- JTable
- JCheckBox
- ButtonGroup
- ActionListener
- AbstractAction
- paintComponent