
	Europaschule Schulzentrum SII Utbremen	<b>Java-Programmierung</b>	05.05.2014 PET	
PP		Datenbank „hsqldb“ - Einführung		Seite 1 von 5

## JAVA – Eine Einführung in Datenbanken I

(modifizierte Version: (Quelle (28.04.14): Java Blog Buch - Ein Buch über Java Programmierung als Blog;  
<http://www.java-blog-buch.de/>)

Es gibt viele Möglichkeiten, Daten aus einem Java-Programm auf die Festplatte zu schreiben und bei einem erneuten Programmstart wieder zu laden. Im Fach der Informatik haben Sie gelernt, wie Sie Textdateien mit Java erstellen, ganze Objekte speichern und die Systemproperties verwenden. In diesem Arbeitsblatt lernen Sie deren Vollendung in der Datenbanktechnik kennen.

### Warum sollte ich Datenbanken verwenden?

Zuerst sei gesagt, dass Sie keinesfalls Datenbanken verwenden *müssen*. Eine Datenbank ist auch nicht für alle Probleme die richtige Lösung, sondern kann falsch eingesetzt den Code sogar komplizieren. Im Grund kann man sagen, dass Datenbanken dann geeignet sind, wenn es um eine große, geordnete Datenmenge geht. Mit „geordnet“ ist hier gemeint, dass alle Daten ein bestimmtes Grundmuster aufweisen, wie zum Beispiel Kundendaten. Als Gegenbeispiel kann man Einstellungsdaten nennen. Diese haben meist verschiedene Eigenschaften, also Datentypen. Es lässt sich schlecht ein Muster finden, und damit ist bei Einstellungsdaten die Datenbank nicht wirklich geeignet. Hier verwendet man entweder die Systemproperties (System und Runtime-Methoden) oder eine XML-Datei.

Auch ist bei der Verwendung von Datenbanken Vorsicht geboten, wenn noch andere Programme auf die Daten zugreifen können.



### Was brauche ich dafür?

Sie benötigen natürlich das JDK, und außerdem noch die Bibliothek HSQLDB. HSQLDB ist eine kostenfreie Java-Datenbank, die einen SQL-Dialekt verwendet. HSQLDB kann über den Link: <http://www.hsqldb.org/> heruntergeladen werden. Die OfficeSuite OpenOffice.org bzw. LibreOffice verwendet ebenfalls HSQLDB.

HSQLDB lässt sich als **Client-Bibliothek** verwenden, es bringt aber auch einen Server mit. In diesem Kapitel werden wir uns vorrangig mit der Client-Bibliothek beschäftigen. HSQLDB verfügt auch über einen eigenen JDBC-Datenbanktreiber, sodass die Einbindung einer Datenbank recht einfach gelingt.

### Grundsätzliches über Datenbanken

Datenbanken zeichnen sich durch eine hochgradig geordnete Struktur aus. Es gibt verschiedene Arten von Datenbanken, in diesem Kapitel werden wir uns mit der einfachsten von ihnen beschäftigen, der **relationalen Datenbank**.

	Europaschule Schulzentrum SII Utbremen	<b>Java-Programmierung</b>	05.05.2014 PET	
PP		Datenbank „hsqldb“ - Einführung	Seite 2 von 5	

Eine **relationale Datenbank** besteht aus einer oder mehreren Tabellen, die wiederum Attribute besitzen. So kann zum Beispiel die Tabelle `HAUS` die Attribute `groesse`, `farbe` und `einwohner` besitzen. Diese Festlegung nennt man *Tabellenschema*. Wenn Sie dieser Aufbau an Ihre erste Begegnung mit Klassen in Java erinnert, liegen Sie richtig.

Häufig lassen sich Tabellen mit Klassen vergleichen. Einem Eintrag in die Tabelle `HAUS` entspräche dann einem Objekt der Klasse `HAUS` in Java. Im Unterschied zu Objekten in Java müssen Sie sich aber bei der Datenbankprogrammierung selbst darum kümmern, dass ein Objekt eindeutig identifizierbar ist. Meistens wird dafür ein Attribute `id` verwendet.

Eine Tabelle kann beliebig viele Einträge annehmen und es können beliebig viele Tabellen in einer Datenbank definiert werden.

Die Daten werden mit sogenannten **SQL-Abfragen** wieder aus der Datenbank geholt. Dabei kann genau definiert werden, welche Information man haben möchte. Auch tabellenübergreifende Abfragen sind dabei möglich.

In `HSQldb` werden Datenbanken über einen sogenannten **JDBC-Treiber** angesprochen. Für uns sieht es aber so aus, als benutzten wir reinen Java-Code, der *Low-Level-Kram* wird von der Bibliothek selbst erledigt.

## 1. SQL-Grundlagen

Hier soll es vor allem um die Verwendung von Datenbanken in Java gehen, nicht darum, wie man Datenbanken allgemein bedient. Es werden zwar die Grundbefehle erklären, doch nur genau so weit, wie es für das Verständnis dieses Kapitels notwendig ist.

Wer SQL mit Java zu 100% ausnutzen möchte, der sollte sich im Internet ein SQL-Tutorial suchen. Ein relativ gutes Tutorial finden Sie unter dem folgenden Link:



SQL-Tutorial: <http://www.1keydata.com/de/sql/>

### 1.2 Datenbankbindung

Im Folgenden werden wir uns einige Einstellungen an der Datenbank selbst anschauen und außerdem zum ersten Mal mit der Datenbank in Verbindung treten.

### 1.3 Eine Verbindung zur Datenbank aufbauen

Nachdem Sie die `hsqldb.jar` in Ihren Classpath (lib) eingefügt haben, können Sie eine Verbindung zu einer Datenbank aufbauen. Wenn die gewünschte Datenbank nicht existiert, wird sie automatisch erstellt. Alle Klassen, die mit der Datenbank zu tun haben, kommen aus dem Paket `java.sql`. Diesen Import schreiben Sie sich am besten gleich in Ihre Quelldatei, denn in diesem Kapitel wird der Import dieses Pakets vorausgesetzt.

	Europaschule Schulzentrum SII Utbremen	<b>Java-Programmierung</b>	05.05.2014 PET	
PP		Datenbank „hsqldb“ - Einführung	Seite 3 von 5	

```
try {
    Class.forName("org.hsqldb.jdbcDriver");
    Connection connection = DriverManager.getConnection("jdbc:hsqldb:file:
C:/jbb/base.db;shutdown=true", "sa", "");
} catch (ClassNotFoundException e) {
    System.err.println("Keine Treiberklasse gefunden.");
}
```

In Zeile 2 wird der Datenbanktreiber geladen und in Zeile 3 die eigentliche Verbindung zur Datenbank hergestellt. Dabei wird der Pfadname zur Datenbank und deren Dateiname selbst angegeben. Die Dateierweiterung ist egal, sie muss nur konsequent verwendet werden. In der URL wird außerdem angegeben, welcher Treiber verwendet werden soll und ob die Datenbank automatisch geschlossen werden soll. Die beiden nächsten Argumente geben den Benutzernamen und das Passwort an. Hier sollten sie zunächst die Standard-Angaben stehen lassen.

Wenn nichts schief ging, können Sie von nun an mit dem `Connection`-Objekt auf die Datenbank lesend und schreibend zugreifen.

#### 1.4 Das Herunterfahren zuerst

Wenn Sie mittels eines Programms auf eine HSQL-Datenbank zugreifen und Änderungen schreiben, werden diese zunächst nur im Speicher gehalten, nicht auf der Festplatte. Damit die Änderungen dauerhaft werden, muss eine `Connection` geschlossen werden. Das passiert mit folgendem Aufruf:

```
try{
    connection.commit();
}catch(SQLException ex)
    ex.printStackTrace();
}
```



Diesen Aufruf haben wir vorgeschoben, damit Sie produktiv mit der Datenbank auch über mehrere Programmneustarts arbeiten können. Rufen Sie diese Methode immer am Ende Ihres Programms auf. Wir werden sie nicht an jedes Listing dieses Kapitels anhängen.

#### 1.5 Tabellen einfügen

Im Paket `java.sql` finden sich die Klassen `Statement` und `PreparedStatement`. Beide generieren und führen SQL-Statements aus. Zum Unterschied der beiden kommen wir später. Wir werden in diesem Kapitel `PreparedStatement` verwenden. Zunächst kann man sich über ein `Connection`-Objekt eine Anweisung vorbereiten:

```
try{
    PreparedStatement ps = connection.prepareStatement("CREATE TABLE user
(id INT, name VARCHAR_IGNORECASE(30), pass VARCHAR(10))");
}catch(SQLException ex){
    ex.printStackTrace();
}
```

Der Datentyp `VARCHAR_IGNORECASE(30)` ist drückt aus, dass in SQL-Abfragen in dieser

	Europaschule Schulzentrum SII Utbremen	<b>Java-Programmierung</b>	05.05.2014 PET	
PP		Datenbank „hsqldb“ - Einführung	Seite 4 von 5	

Spalte nicht auf die Groß/Kleinschreibung geachtet wird. Gespeichert wird sie trotzdem.

Der Query wird folgendermaßen ausgeführt:

```
try{
    ps.executeUpdate();
}catch(SQLException ex){
    ex.printStackTrace();
}
```

Auf dieselbe Art und Weise werden alle SQL-Befehle behandelt, die die Daten der Datenbank *verändern*. Daher wird in diesem Kapitel nicht näher auf `INSERT`, `UPDATE` usw. eingegangen.



`executeUpdate()` gibt zurück, wie viele Zeilen von der Änderung betroffen waren.

## 1.6 SQL-Abfragen

Neben dem Einfügen der Daten müssen diese auch wieder aus der Datenbank geholt werden. Dies passiert zwar ebenfalls mit einem `(PreparedStatement)`, doch die Ergebnisse werden in einer anderen Form präsentiert. Angenommen, sie wollen aus der Datenbank alle Benutzer filtern, deren Namen gleich “Max Mustermann” ist. Die dazugehörige SQL-Abfrage übergeben sie wie gewohnt der `prepareStatement(...)`-Methode. Diesmal rufen Sie aber nicht `executeUpdate()` auf, sondern `executeQuery()`. Diese Methode gibt ein `ResultSet` zurück, welches mit einer `while`-Schleife durchlaufen werden muss:

```
try{
    PreparedStatement ps = connection.prepareStatement("SELECT * FROM user
    WHERE name='Max Mustermann'");
    ResultSet set = ps.executeQuery();
    while(set.next()){
        System.out.println("Name: " + set.getString(1) + ", Passwort: " +
        set.getString("pass"));
    }
}catch(SQLException ex){
    ex.printStackTrace();
}
```

`ResultSet.next()` macht zwei Dinge: Erstens wird überprüft, ob noch ein Eintrag vorhanden ist und dementsprechend `true` oder `false` zurückgegeben. Zweitens wird der interne Zähler des `ResultSet` auf den nächsten Eintrag geschoben. Falls Sie, nachdem `next()` `false` zurückgegeben hat, noch auf die Daten zugreifen wollen, erfolgt eine `Exception`.

	Europaschule Schulzentrum SII Utbremen	<b>Java-Programmierung</b>	05.05.2014 PET	
PP		Datenbank „hsqldb“ - Einführung	Seite 5 von 5	

Wie Sie sehen, bietet das `ResultSet` zwei verschiedene Methoden zum Ermitteln eines Strings an. Zu einen können Sie über den Index der Spalte auf die Daten zugreifen, zum anderen über deren Namen. Bei ersterem müssen Sie beachten, dass die Zählung bei 1 beginnt. Bei letzterem wird der Name verwendet, den Sie in der `SELECT`-Anweisung angegeben haben. Wenn Sie also `SELECT pass AS password ...` festgelegt haben, wird die Spalte unter `password` gefunden. Neben der `getString`-Methode gibt es auch Methoden für die primitiven Datentypen (z.B.: `getInt()`) und Basisklassen (z.B.: `getDate()`). Alle Methoden bieten beide Zugriffsmöglichkeiten.

### 1.7 Schließen der Statements und ResultSets

Vergessen Sie nicht, Ihre `PreparedStatement` und `ResultSet` zu schließen, bevor die Methode beendet wird, damit die JDBC-Ressourcen freigegeben werden. Auch sei hier nochmal erwähnt, dass die `Connection`-Objekte ebenfalls geschlossen (mit `commit()`) werden müssen, bevor die Anwendung beendet wird, damit die Daten permanent in der Datei gespeichert werden.