

Datenbanken & Informationssysteme Übungen Teil 1

Programmierung von Datenbankzugriffen

1. Daten lesen mit JDBC

Schreiben Sie eine Java-Anwendung, die die Tabelle Books in der Datenbank azamon ausgibt. Verwenden Sie dabei die SQL-Anweisung `select * from Books`. Angezeigt werden soll eine Kopfzeile mit den Namen der in der Tabelle enthaltenen Spalten, sowie der Inhalt der Tabelle.

Ein SQL-Skript für das Anlegen der Datenstruktur und das Befüllen der Tabellen zur Datenbank azamon finden Sie hier:

<https://homepages.thm.de/~hg11260/mat/azamon-create.sql>. Das Skript ist für PostgreSQL, können aber für jedes andere SQL-Datenbankmanagementsystem nach eventuellen Anpassungen verwendet werden.

2. Daten lesen mit der Java Persistence API

Schreiben Sie ein Java-Programm, das die Bücher in der Tabelle Books ausgibt. Angezeigt werden sollen die Inhalte der Zeilen der Tabelle.

Sie benötigen eine Implementierung der JPA, zu finden z.B. EclipseLink unter <http://www.eclipse.org/eclipselink/>.

3. Ein erster Versuch mit JDBC

Eine Softwareentwicklerin oder ein Softwareentwickler hat sein erstes JDBC-Programm geschrieben. Es soll einfach die erste Spalte der ersten Zeile in der Tabelle Books in der Datenquelle azamon ausgeben.

```
import java.sql.*;
public class FirstJDBC {
    public static void main(String[] args) {
        try {
            Class.forName("org.postgresql.Treiber");
            Connection con = DriverManager.getConnection(
                "jdbc:postgresql:azamon","x","x");
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("select * from Books");
            System.out.println(rs.getString(0));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Beim Testen des Programms treten Exceptions auf. Klären Sie warum und geben Sie an, was man verbessern muss:

- (a) `java.lang.ClassNotFoundException: org.postgresql.Treiber`
Woran liegt es?
- (b) Nachdem der Fehler korrigiert ist, tritt folgendes auf:
`java.sql.SQLException: Ungültiger Cursorstatus`
Woran liegt's?
- (c) Doch nachdem dieser Fehler korrigiert ist, tritt noch einer auf:
`java.sql.SQLException: Ungültiger Deskriptorindex`
 Klären Sie auch hier den Grund für die Exception auf.

4. Parametrisierte Anweisung

- (a) Schreiben Sie ein JDBC-Programm, mit dem man in der Tabelle Books von amazon herausfinden kann, wieviele Bücher eines Autors verzeichnet sind. Das Programm soll so gestaltet sein, dass man interaktiv immer wieder neue Autorennamen eingeben kann. (Tipp: wir brauchen keine komplette GUI, es reicht für die Eingabe `showInputDialog` von Java Swing und für die Ausgabe die Konsole zu verwenden.)
- (b) Schreiben Sie ein JDBC-Programm, mit dem man ein neues Buch in die Tabelle eingeben kann. Verwenden Sie dabei eine parametrisierte Anweisung. (Auch dieses Programm muss nicht unbedingt eine GUI haben.)

5. SQL-Injection

- (a) Schreiben Sie ein JDBC-Programm, mit dem man nach einem Buch in der Tabelle Books suchen kann, so dass es einem Angreifer gelingt, dieses Programm zu verwenden, um einen Eintrag in der Tabelle zu löschen.
- (b) Wie müssen Sie das Programm ändern, um diesen Angriff zu vereiteln?

6. Java Sicherheits-Richtlinien

In den Java Coding Guidelines von CERT und SEI hat die erste Regel die Überschrift „IDS00-J. Prevent SQL injection“, zu finden hier: <https://www.securecoding.cert.org/confluence/display/java/IDS00-J.+Prevent+SQL+injection>

Vollziehen Sie das Beispiel zur Regel in einem eigenen Java-Programm nach.

7. Analyse eines JDBC-Tabellen-Editors

Analysieren Sie den Aufbau eines Tabellen-Editors, den Mitarbeiter von IBM geschrieben haben:

<http://www.ibm.com/developerworks/data/library/techarticle/0206chaitgal/0206chaitgal.html>

- (a) Beschreiben Sie, welche Mechanismen von JDBC in diesem Programm verwendet werden.

- (b) Diskutieren Sie, welche grundlegenden Unzulänglichkeiten dieses Programm hat. Nennen Sie dazu mindestens 3 Punkte.

8. Metainformationen mit JDBC

In JDBC gibt es zwei Interfaces für Metainformationen:
`DatabaseMetaData` und `ResultSetMetaData`.

Beschreiben Sie jeweils, welche Informationen über diese Interfaces bereitgestellt werden und geben Sie an, welchen *Konzepten* einer SQL-Datenbank diese Interfaces entsprechen.

- (a) `DatabaseMetaData`
- (b) `ResultSetMetaData`

9. DBMS-unabhängige Programmierung in JDBC und ADO.NET

Man möchte APIs zum Zugriff auf Datenbankmanagementsysteme oft gerne so verwenden, dass man möglichst unabhängig vom jeweiligen DBMS ist: das übersetzte, ausführbare Programm soll mit jedem beliebigen DBMS verwendet werden können. Erläutern Sie die grundlegenden Konzepte.

- (a) Welchen Mechanismus sieht JDBC vor, um die Unabhängigkeit vom DBMS zu erreichen?
- (b) Welchen Mechanismus sieht ADO.NET (ab Version 2.0) vor, um die Unabhängigkeit vom DBMS zu erreichen?
- (c) Worauf muss man außer den in (a) und (b) genannten Mechanismen achten, um Unabhängigkeit vom DBMS zu erreichen?

10. Datenbankschema mit JPA erzeugen

Man kann in JPA aus dem Objektmodell, definiert in den Entitätstypen als Java-Klassen, automatisch ein Datenbankschema erzeugen.

Abbildung 1 zeigt das Objektmodell zur Beispieldatenbank *Employees Sample Database* von MySQL als Klassendiagramm der UML.

Implementieren Sie dieses Objektmodell in Java und verwenden Sie die JPA um damit ein Datenbankschema zu erzeugen. Vergleichen Sie das Ergebnis mit dem Datenbankschema der *Employees Sample Database*. Ein Diagramm des Datenbankschemas findet man auf <https://dev.mysql.com/doc/employee/en/sakila-structure.html>, es ist aber nicht ganz korrekt. Das SQL-Skript zum Erzeugen und Füllen der Beispiel-Datenbank steht auf https://github.com/datacharmer/test_db.

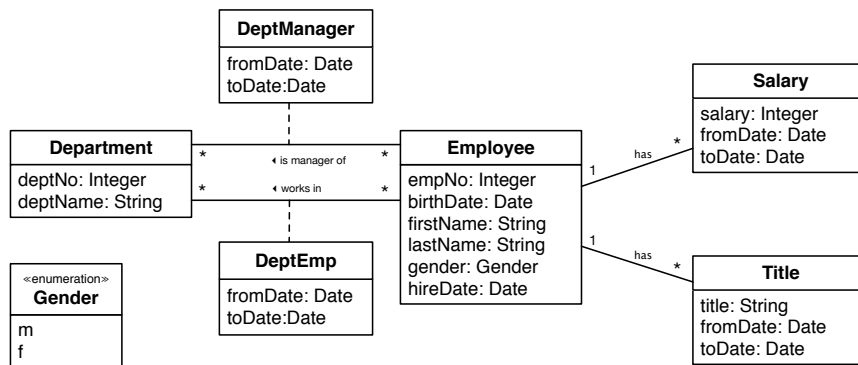


Abbildung 1: Objektmodell zur *Employees Sample Database*