Praktikum JDBC & File Input / Output

Damit Sie im JDBC-Praktikum keine Zeit verlieren, um die Datenbank und JDBC einzurichten, sollten Sie diese vor dem Praktikum installieren und konfigurieren.
Danach werden Sie zwei aufeinander aufbauende Übungen ausführen um den grundlegenden Umgang mit JDBC und File IO, sowie die Anwendung von Data Access Objects (DAO) zu üben.

1. Vorbereitung Praktikum JDBC & File Input/Output

Einrichten der Datenbank

Die Datenbank und Tabellen, die Sie hier einrichten, werden auch in weiteren Praktika wiederverwendet. Deshalb richten Sie die Datenbank am einfachsten auf dem DB-Laborserver Dublin ein. Dadurch brauchen Sie auch keinen lokalen DB- und Webserver auf Ihrem eigenen Rechner.
Die Anleitungen zum Einrichten der DB-Laborserver Umgebung finden Sie hier im SoE-Intranet:
https://intra.zhaw.ch/departemente/school-of-engineering/services/services-school-of-engineering/informatik-soe/it-umgebung-school-of-engineering/db-laborserver.html
(Den Link finden Sie auch direkt auf der Hauptseite von Dublin https://dublin.zhaw.ch)

Aufgaben:

a) **Richten Sie sich Ihre PostgreSQL Datenbank ein**
PostgreSQL deshalb, da es die einzige ist, die auch übers interne ZHAW-Netz (bzw. via VPN) direkt angesprochen werden kann, was wir später brauchen werden.
Die Anleitung finden Sie im Bereich „Einrichten und Zugriff auf PostgreSQL“.
- Sie benötigen SSH Zugang (Links zu PuTTY für Windows User auf obiger Intranetseite in der rechten Spalte)
- Verwenden Sie als DB-Passwort NICHT Ihr ZHAW Passwort!

b) **Erstellen Sie die Tabelle „picture“ in Ihrer Datenbank**
Das SQL-Script dazu finden Sie am Ende von Kapitel 1 oder im ZIP-Archiv zu dieser Übung im Labs-Ordner auf OLAT.
Sie können das SQL-Script von Kommandozeile auf Dublin mit `psql -f init-picture.sql` oder einem beliebigen Tool wie z.B. pgAdmin3 oder dem Webfrontend phpPgAdmin ausführen. Verifizieren Sie, dass die Tabelle existiert und die Werte enthalten sind.

Einrichten JDBC Umgebung

Damit Sie im Praktikum direkt loslegen können benötigen Sie die JDBC-Treiber und müssen Ihre Entwicklungsumgebung bzw. Projekt konfigurieren.

Aufgaben:

a) **JDBC-Treiber laden und in Projekt einbinden**
Laden Sie den passenden JDBC-Treiber von http://jdbc.postgresql.org/download.html. Für Java 7/8 nehmen Sie die aktuelle JDBC41-Version (für Java 6 die JDBC4-Version). Diese ist rückwärtskompatibel mit der älteren PostgreSQL-Version des Servers.
Erstellen Sie ein neues Projekt fürs Praktikum und fügen Sie den Treiber zum classpath hin-

zu. In Eclipse kopieren Sie die JAR-Datei z.B. in ein Projektverzeichnis `lib` und fügen es über *Project* > *Properties* > *Libraries* > *Add JARs...* zum Pfad hinzu.

b) **Testen Sie die JDBC-Verbindung zur Datenbank mit der Klasse `TestJdbcConnection`**
Erstellen/Importieren Sie die Klasse `TestJdbcConnection` aus dem Anhang bzw. dem ZIP-Archiv im Labs Ordner auf OLAT.
Führen Sie die Klasse aus und testen Sie ob die Verbindung klappt.
Wichtig: Nur im ZHAW-Netzwerk bzw. via VPN-Verbindung möglich.
Was fällt Ihnen bezüglich Exception-Handling auf?

Files

Datei `init-picture.sql`:

```
-- delete old table if it exits
drop table if exists picture;

-- create table 'picture'
create table picture (
    id          serial       PRIMARY KEY,
    date        timestamp    DEFAULT CURRENT_TIMESTAMP,
    longitude   float        DEFAULT NULL,
    latitude    float        DEFAULT NULL,
    altitude    float        DEFAULT NULL,
    title       varchar(200) DEFAULT NULL,
    comment     text          DEFAULT NULL,
    url         varchar(200) DEFAULT NULL
) without OIDS;

-- fill table 'picture' with some demo data
INSERT INTO picture (date, longitude, latitude, altitude, title, comment, url)
VALUES
  ('2014-02-25 00:00:00',47.12,  8.12,  422.12, 'kja','sad',
'http://endingcampo.files.wordpress.com/2011/10/coder__smaller__by_dwerg85.png'),
  ('2014-02-26 00:00:00',123.12, 8.12,  422.12, 'df','asdjo',
'http://endingcampo.files.wordpress.com/2011/10/coder__smaller__by_dwerg85.png'),
  ('2014-02-27 00:00:00',47.1233,8.12,  422.12, 'coder','in progess',
'http://endingcampo.files.wordpress.com/2011/10/coder__smaller__by_dwerg85.png'),
  ('2014-03-25 00:00:00',48.2134,8.2413,422.12, 'code monkey','a funny one',
'http://alltech-nologic.com/wp-content/uploads/2012/02/code-monkey.jpg'),
  ('2014-04-01 00:00:00',47.12,  8.12,  422.12, 'code monkey','monkey',
'http://alltech-nologic.com/wp-content/uploads/2012/02/code-monkey.jpg'),
  ('2014-04-02 00:00:00',33.33,  22.22, 422.12, 'test1','khbvlkwpöjpbdvpi',
'http://endingcampo.files.wordpress.com/2011/10/coder__smaller__by_dwerg85.png');
```

Datei: `TestJdbcConnection.java`

```
import java.sql.*;
import java.io.*;

public class TestJdbcConnection {

    public static void main(String args[])
    throws SQLException, IOException
    {
        String dbUser = prompt("User: ");
        String dbPass = prompt("Passwort: ");
        String dbURL = "jdbc:postgresql://dublin.zhaw.ch/" + dbUser;

        try (Connection con = DriverManager.getConnection(dbURL, dbUser, dbPass)) {
            Statement stmt = con.createStatement();
            ResultSet rset = stmt.executeQuery("select * from picture");
            int numCols = rset.getMetaData().getColumnCount();
            PrintWriter out = new PrintWriter(System.out, true);
            for (int i = 1; i <= numCols; i++) {
                out.print(rset.getMetaData().getColumnLabel(i) + "\t");
            }
            out.println();
            while (rset.next()) {
                for (int i = 1; i <= numCols; i++) {
                    out.print(rset.getObject(i) + "\t");
                }
                out.println();
            }
        } catch (SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
            System.exit(-1);
        }
    }
    // prompt function -- to read input string
    static String prompt(String prompt) {
        try {
            StringBuffer buffer = new StringBuffer();
            System.out.print(prompt);
            System.out.flush();
            InputStreamReader in = new InputStreamReader(System.in);
            int c = in.read();
            while (c != '\n' && c != -1) {
                buffer.append((char) c);
                c = in.read();
            }
            return buffer.toString().trim();
        } catch (IOException e) {
            return "";
        }
    }
}
```

2. Picture Import

In dieser Übung verwenden Sie die grundlegenden JDBC und File IO Funktionen von Java um Daten von der Kommandozeile bzw. den Inhalt einer Datei in eine Datenbank zu importieren.

Im ZIP-Archiv zu diesem Praktikum finden Sie das Programm `PictureImport.java`. In `main()` werden die folgenden drei Funktionen nacheinander aufgerufen:

- `createPicture()`: Daten von Kommandozeile einlesen und Picture Objekt erzeugen
- `addPicture()`: Picture Objekt in Datenbank hinzufügen
- `getPicture()`: Picture Objekt aus Datenbank auslesen

Die Methode `createPicture()` ist schon fertig implementiert und ermöglicht Ihnen die Daten eines Bildes, welche im Picture Objekt gespeichert werden auf der Kommandozeile einzugeben.

Aufgaben:

a) Implementieren Sie die Methoden `addPicture()` und `getPicture()`, welche die Daten in die von Ihnen vorbereitete Datenbank auf Dublin speichern bzw. auslesen.
In `addPicture()` soll automatisch auch die `id` des Datensatzes ausgelesen und im Picture Objekt gespeichert werden. Verwenden Sie dazu die im Unterricht (DAO insert-Methode) gezeigte Technik mit `Statement.`*`RETURN_GENERATED_KEYS`*`.`
Tipp: Das `java.util.Date` Objekt müssen Sie in ein `java.sql.Date` Objekt umwandeln (und umgekehrt beim Auslesen). Sie können dazu den Time-Wert (Sekunden seit 1.1.1970) des Date-Objektes verwenden:
`java.sql.Date sqlDate = new java.sql.Date(anyUtilDate.getTime());`

b) Erweitern Sie das Programm um die Methode `importFile(File file)`, welcher Sie eine Datei mit Datensätzen übergeben, die in die Datenbank importiert werden sollen. Die Datei soll im Character-Separated-Value (CSV) Format vorhanden sein. Dabei ist jede Zeile ein Datensatz (in unserem Fall ein Picture) und die einzelnen Felder werden durch ein Trennzeichen/Delimiter (in unserem Fall „;') voneinander separiert. Als Beispiel können Sie die mitgelieferte Datei `data.csv` verwenden.
Vorgehen:
Lesen Sie die Datei zeilenweise ein, trennen Sie die einzelnen Felder auf z.B. mit der Funktion `String.split()` und füllen Sie die einzelnen Felder in ein Picture Objekt ab, das Sie dann mit `addPicture()` speichern können.

3. Picture Data Access Objects

Data Access Objects (DAOs) separieren die Geschäftslogik Ihrer Anwendung von der Datenhaltung (Persistenzschicht) und ermöglichen den Wechsel zwischen verschiedenen Speichertechnologien, ohne dass die Anwendungslogik angepasst werden muss.

In dieser Übung werden Sie DAO-Klassen für Picture erstellen und testen.

Im ZIP-Archiv zu diesem Praktikum finden Sie im Ordner `dao` die vorbereiteten Klassen:

- `ReadOnlyDAO` ist ein generisches Interface mit den Grundfunktionen zum Lesen der Daten.

- `WritableDAO` erweitert ReadOnlyDAO um die Methoden die schreibende Zugriffe benötigen.
- `PictureDAO` ist das spezifische DAO Interface für das Picture Objekt. Dieses Interface wird von der Geschäftslogik Ihrer Anwendung verwendet. Es erweitert `WritableDAO` und erweitert dies zusätzliche Methoden, die nur für Picture relevant sind.
- `PictureJdbcDAO` ist die Implementierung für die Speicherung der Daten in der Datenbank
- `PictureFileDAO` ist die Implementierung für die Speicherung der Daten in Dateien

Aufgaben:

a) Implementieren Sie die Klasse `PictureJdbcDAO`.
Beginnen Sie mit den Methoden, die Sie in Aufgabe 1 schon implementiert haben (`addPicture` → `insert`; `readPicture` → `findById`) und ergänzen Sie auch die restlichen Funktionen (`update`, `delete`, `findAll`, ...)

b) Erstellen Sie ein Testprogramm, in welchem Sie die DAO Methoden testen können. Zum Beispiel können Sie mit dem Hauptprogramm aus Übung 1 starten und dieses so umbauen, dass es die Ihr DAO-Objekt verwendet. Erweitern Sie dieses, dass auch die anderen Funktionen getestet werden.

c) Implementieren Sie die Klasse `PictureFileDAO`
Als Dateiformat soll das in Übung 1 besprochene CSV-Format dienen. Dieses müssen Sie evtl. um benötigte Felder (z.B. id) erweitern. Auch hier beginnen Sie mit der bereits vorhandenen Logik aus Übung 1 (`importFile` → `findAll`) und implementieren als erstes die lesenden Funktionen (`count`, `findById`, `findByPosition`).
Erweitern Sie das Testprogram um diese Funktionen zu testen.

d) Implementieren Sie die Methode `insert`; Anhängen eines Datensatzes am Ende der Datei. Wichtigste Frage ist, wie erzeugen Sie die `id`? Wie merken Sie sich bzw. berechnen Sie die nächste `id`? Eine Variante wäre, die erste Zeile der Datei für solche Metainformationen zu verwenden (eine Art Header). Andere Ideen?

e) Implementieren Sie die restlichen Funktionen (`delete` und `update`)
Diese sind etwas aufwändiger, da Dateien sequentielle Medien sind und nicht einfach Zeichen mittendrin eingefügt bzw. gelöscht werden können. Wie kann man das lösen? Funktioniert Ihre Variante auch mit sehr grossen Dateien (grösser als Hauptspeicher des Rechners)?