JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll

Prepared Statements

INSY
4AHITT 2015/16

Matthias Stickler, Antonio Pavic

Version 1.2

Note:

Betreuer: M. Borko

Begonnen am 25. Mai 2016

Beendet am 26. Mai 2016

Inhaltsverzeichnis

1 Einführung

PreparedStatements sind in JDBC eine Möglichkeit SQL-Befehle vorzubereiten um SQL-Injections zu vermeiden. Die Typüberprüfung kann somit schon bei der Hochsprache abgehandelt werden und kann so das DBMS entlasten und Fehler in der Businesslogic behandelbar machen.

1.1 Ziele

Es ist erwünscht Konfigurationen nicht direkt im Sourcecode zu speichern, daher sollen Property-Files [3] zur Anwendung kommen bzw. CLI-Argumente (Library verwenden) [1,4] verwendet werden. Dabei können natürlich Default-Werte im Code abgelegt werden.
Das Hauptaugenmerk in diesem Beispiel liegt auf der Verwendung von PreparedStatements [2]. Dabei sollen alle CRUD-Aktionen durchgeführt werden.

1.2 Quellenangaben

[1] Apache Commons CLI; Online:
http://commons.apache.org/proper/commons-cli/
[2] Java Tutorial JDBC "Prepared Statements"; Online:
https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html
[3] Java Tutorial Properties; Online:
https://docs.oracle.com/javase/tutorial/essential/environment/properties.html
[4] Overview of Java CLI Libraries; Online:
http://stackoverflow.com/questions/1200054/java-library-for-parsing-command-line-parameters

1.3 Aufgabenstellung

Verwenden Sie Ihren Code aus der Aufgabenstellung "Simple JDBC Connection" um Zugriff auf die Postgresql Datenbank "Schokofabrik" zur Verfügung zu stellen. Dabei sollen die Befehle (CRUD) auf die Datenbank mittels PreparedStatements ausgeführt werden. Verwenden Sie mindestens 10000 Datensätze bei Ihren SQL-Befehlen. Diese können natürlich sinnfrei mittels geeigneten Methoden in Java erstellt werden.

Die Properties sollen dabei folgende Keys beinhalten: host, port, database, user, password

2 Ergebnisse

2.1 Verbindung zur Datenbank

Wie bei der JDBC Connection wird der PostgreSQL JDBC driver verwendet. Das Aufbauen einer Verbindung benötigt ein PGSimpleDataSource Objekt. In diesem werden dann Servername, Datenbankname, Username, Passwort und Port gesetzt. Das alles passiert in der stickler.DBConnect Klasse.

```
//DB Connection
ds = new PGSimpleDataSource();
ds.setServerName("192.168.248.135");
ds.setDatabaseName("schokodb");
ds.setUser("schokouser");
ds.setPassword("root");
ds.setPortNumber(5432);
```

Anschließend kann die Verbindung durch connect() aufgebaut bzw. close() geschlossen werden. Dies verlangt jedoch Exceptionhandling.

connect():

```
try{
    connection = ds.getConnection();
}catch (SQLException e){
    System.err.println("Connection failed!");
    e.printStackTrace(System.err);
}
```

close():

```
try{
    connection.close();
}catch (SQLException e){
    System.err.println("Closing Connection failed!");
    e.printStackTrace(System.err);
```

2.2 Prepared Statements

Zunächst Wird eine Wrapper Methode benötigt um die Prepared Statements auf die vorhandene Connection zu setzen.

```
public PreparedStatement preparedStatement(String statement) {
    try {
        return connection.prepareStatement(statement);

    } catch (SQLException e) {
        e.printStackTrace(System.err);
    }
    return null;
}
```

In der stickler.PreparedStatements Klasse wird nun ein PreparedStatement Attribut gesetzt.

```
private static PreparedStatement ps;
```

Dieses soll als globaler Speicherort der Prepared Statements aus den CRUD Methoden der Klasse fungieren.

CREATE

CRUD Create wird durch eine einfache INSERT-Operation der Tabelle produkt realisiert.

Die insert(DBConnect, int, String, int) Methode benötigt 4 Parameter. Eine Connection, die wir in der DBConnect Klasse aufgebaut haben und 3 weitere Werte, die die Tabelle befüllen sollen.

Zunächst wird unser globales Attribut mit einem unvollständigen SQL-Statement, gefüllt. Hierbei ist „?" ein Platzhalter für anschließende Vervollständigung des Statements. Die Vervollständigung der Statements benötigt wieder Exceptionhandling.

```
public static void insert(DBConnect connection, int number, String value, int weight) {
    ps = connection.preparedStatement("INSERT INTO produkt Values(?,?,?);");
    try {
        ps.setInt(1, number);
        ps.setString(2, value);
        ps.setInt(3, weight);
        ps.execute();
    } catch (SQLException e) {
        System.err.println("Update failed!");
        e.printStackTrace(System.err);
    }

}
```

Zuletzt wird in der main() Methode getestet.

```
System.out.println("Inserting into produkt...");
insert(con, 150, "TestSchokolade", 455);
insert(con, 151, "tobeDeleted", 777);
```

READ

CRUD Create wird durch eine einfache SELECT-Abfrage der Tabelle produkt realisiert.

Die select(DBConnect, int) Methode benötigt 2 Parameter. Eine Connection, die wir in der DBConnect Klasse aufgebaut haben und 1 weiterer Wert, der für das auswählen der gewünschten Daten benötigt wird.

Zunächst wird unser globales Attribut mit einem unvollständigen SQL-Statement, gefüllt. Hierbei ist „?" ein Platzhalter für anschließende Vervollständigung des Statements. Die Vervollständigung der Statements benötigt wieder Exceptionhandling.

```
public static void select(DBConnect connection, int weight) {
    ps = connection.preparedStatement("SELECT * FROM produkt WHERE gewicht < ?;");
    try {
        ps.setInt(3, weight);
        ps.execute();
    } catch (SQLException e) {
        System.err.println("Update failed!");
        e.printStackTrace(System.err);
    }
}
```

Zuletzt wird wieder getestet.

```
System.out.println("Selecting produkt...");
select(con, 455);
```

UPDATE

CRUD Create wird durch eine einfache UPDATE-Operation der Tabelle produkt realisiert.

Die update(DBConnect, int, String) Methode benötigt 3 Parameter. Eine Connection, die wir in der DBConnect Klasse aufgebaut haben und 2 weitere Werte, die zur Auswahl der gewünschten Daten und zur Veränderung der Daten benötigt werden.

Zunächst wird unser globales Attribut mit einem unvollständigen SQL-Statement, gefüllt. Hierbei ist „?" ein Platzhalter für anschließende Vervollständigung des Statements. Die Vervollständigung der Statements benötigt wieder Exceptionhandling.

```
public static void update(DBConnect connection, int number, String value) {
    ps = connection.preparedStatement("UPDATE produkt SET bezeichnung = ? WHERE nummer = ?;");
    try {
        ps.setInt(1, number);
        ps.setString(2, value);
        ps.execute();
    } catch (SQLException e) {
        System.err.println("Update failed!");
        e.printStackTrace(System.err);
    }

}
```

Zuletzt wird wieder getestet.

```
System.out.println("Updating produkt...");
update(con, 150, "UpdatedTestSchokolade");
```

DELETE

CRUD Create wird durch eine einfache DELETE-Operation der Tabelle produkt realisiert.

Die update(DBConnect, int) Methode benötigt 3 Parameter. Eine Connection, die wir in der DBConnect Klasse aufgebaut haben und 1 weiterer Wert, der zur Auswahl gewünschten Daten benötigt wird.

Zunächst wird unser globales Attribut mit einem unvollständigen SQL-Statement, gefüllt. Hierbei ist „?" ein Platzhalter für anschließende Vervollständigung des Statements. Die Vervollständigung der Statements benötigt wieder Exceptionhandling.

```
public static void delete(DBConnect connection, int number) {
    ps = connection.preparedStatement("DELETE FROM produkt WHERE nummer = ?;");
    try {
        ps.setInt(1, number);
        ps.execute();
    } catch (SQLException e) {
        System.err.println("Update failed!");
        e.printStackTrace(System.err);
    }
}
```

Zuletzt wird wieder getestet.

```
System.out.println("Deleting produkt...");
delete(con, 151);
```

3 Versionisierung

Über Github wird die Versionisierung verwaltet:

https://github.com/matthiasstickler/Prepared-Statements

4 Zeitaufzeichnung

AUFGABE	ANTONIO PAVIC	MATTHIAS STICKLER
JAVA CODE	0h	4h
RECHERCHE	2h	1h
DOKUMENTATION	1h	2h
GESAMT	3h	7h