JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll

PreparedStatements

INSY/SEW
4AHITM 2015/16

Matthias Mischek
Benedikt Berger

Version 1.0

Note:

Betreuer:

Begonnen am 1. April 2016

Beendet am 6. 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.

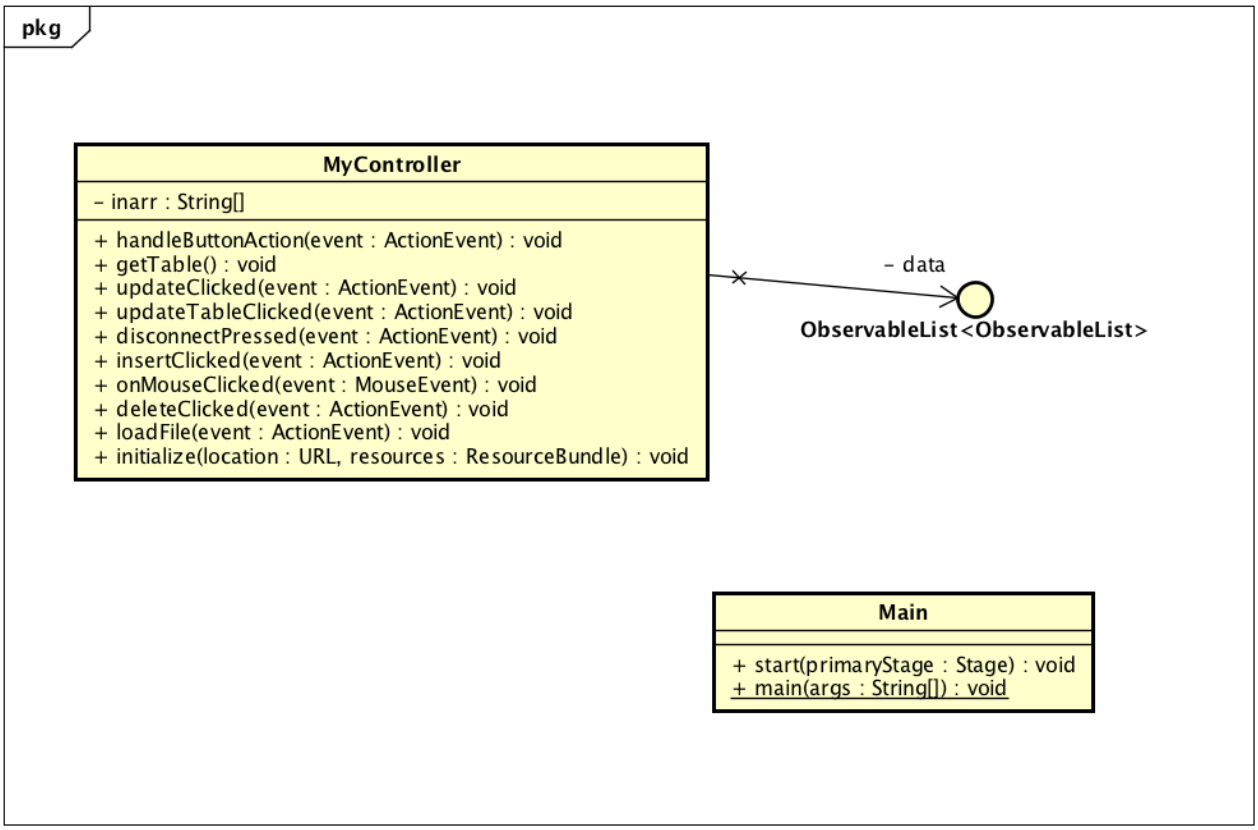
1.2. Aufgabenstellung

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
Vergessen Sie nicht auf die Meta-Regeln (Dokumentation, Jar-File, etc.)! Die Testfälle sind dabei zu ignorieren. Diese Aufgabe ist als Gruppenarbeit (2 Personen) zu lösen.
[BOR16]

2. Vorbereitungen

2.1. Designüberlegung

Bei diesem Beispiel könnten wir einiges von dem Fussballverein Beispiel übernehmen. Die Struktur des Java-Projektes wird in eine Main.java, eine MyController.java und ein MyView.fxml aufgeteilt. In der Main.java werden die Dateien gemeinsam zusammengeführt und ausgeführt. MyController.java enthält die Logistik, wo auch auf ActionEvents reagiert wird. Zuletzt MyView.fxml welches den Aufbau der GUI enthält.



2.1. JDBC

In diesem Beispiel ist es möglich die Daten für die Verbindung dynamisch über die GUI einzugeben oder über ein File auszulesen.

[ORA01]

```
public void connect(ActionEvent event) throws SQLException {

	data = FXCollections.observableArrayList();

	// Datenquelle erzeugen und konfigurieren
	ds = new PGSimpleDataSource();
	ds.setServerName(ip.getText());
	ds.setDatabaseName(dbName.getText());
	ds.setUser(user.getText());
	ds.setPassword(passwort.getText());
	// Verbindung herstellen
	try

	{
		this.con = ds.getConnection(); // Verbinden

		updatePane.setDisable(false);
		anzeigePane.setDisable(false);
		insertPane.setDisable(false);
		updateSpielT.setDisable(false);
		anzeigeSpielT.setDisable(false);
		status.setText("Erfolgreich Verbunden!");

		// Exception handling
	} catch (PSQLException e) {
		System.err.println("Error");
		status.setText("Bitter erneut versuchen.");

	} catch (Exception se) {
		con.rollback();
		System.err.println("Error");
		status.setText("Bitte erneut versuchen.");
	}

}
```

3. Ergebnisse

3.1. CREATE

```
String sql = "INSERT INTO produkt VALUES (?, ?, ?)";


PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, num);
statement.setInt(2, bez);
statement.setInt(3, gew);

statement.executeUpdate();
```

1.0 Erstellen eines Prepared Statements

Ein Prepared Statement wird wie folgt erstellt:
Mittels setString oder setInt werden die Fragezeichen mit sinnvollen Werten gefüllt.

3.1. UPDATE

Prepared Statement für die UPDATE Operation:

```
String sql = "UPDATE produkt SET bezeichnung=?, gewicht=? WHERE nummer=?";


PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, bez);
statement.setInt(2, gew);
statement.setInt(3, num);

statement.executeUpdate();
```

1.1 Prepared Statement für UPDATE Operation

3.1. DELETE

Prepared Statement für die DELETE Operation:

```
String sql = "DELETE FROM produkt WHERE nummer=?";

PreparedStatement statement = connection.prepareStatement(sql);
statement.setInt(1, Integer.parseInt(inarr[0]));
statement.executeUpdate();
```

1.2 Prepared Statement für DELETE Operation

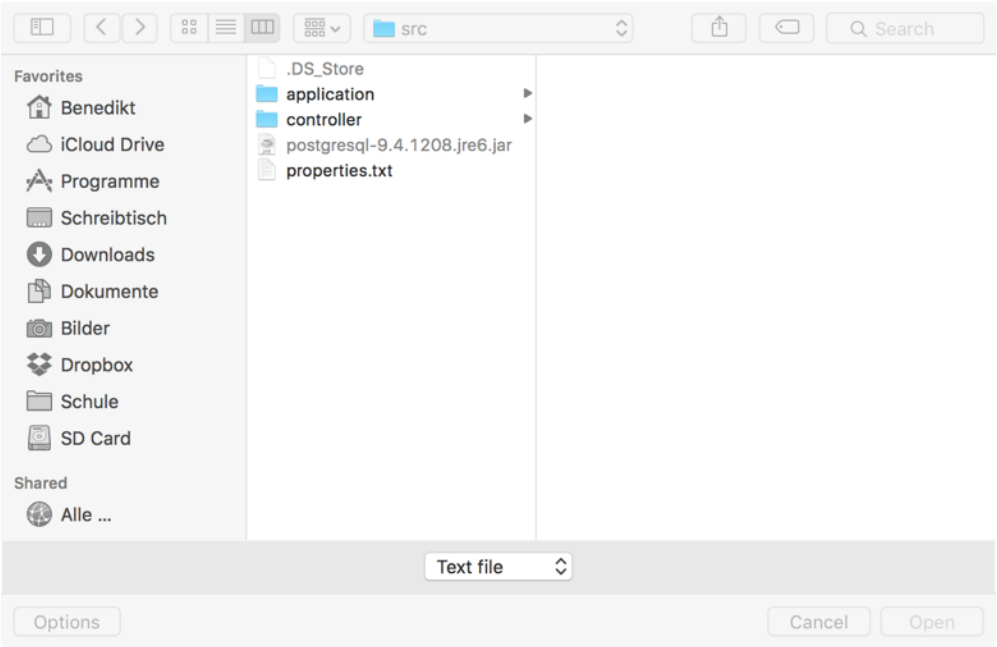
4. Code Erklärung

4.1. File Chooser

Mithilfe des „FileChooser's" können User auch von der GUI aus, .txt Dateien in das Programm einfügen, welche die notwendigen Einstellungen wie IP-Adresse usw. enthalten. Hierfür wird allerdings eine fixe Struktur vorgeschrieben. Hier ein Beispiel eines Properties Files:

```
host:
192.168.0.23
database:
schokofabrik
user:
schokoUser
password:
schokoUser
```

4.1 Properties File für Datenbank Login



Jetzt kann in der GUI durch einen Klick auf „Properties Datei laden“ eine Datei ausgewählt werden, welche bei bestätigen in die Textfelder des Anmeldefensters automatisch eingefügt werden und der User nur noch auf „Verbinden“ klicken muss. Hier ist ein Codeausschnitt des „FileChooser's“ mit dem Filter (damit nur Text Dateien ausgewählt werden können) und dem anschließenden Auslesen der einzelnen Zeilen mithilfe von br.readLine() :

```
FileChooser chooser = new FileChooser();
ExtensionFilter sef = new ExtensionFilter("Text file","*.txt");
chooser.getExtensionFilters().addAll(sef);
chooser.setTitle("Select text file...");
File selectedDirectory = chooser.showOpenDialog(null);

FileReader fr;
try{
    fr = new FileReader(selectedDirectory.getAbsolutePath());
} catch(NullPointerException e2){
    fr = new FileReader("");
}

BufferedReader br = new BufferedReader(fr);
br.readLine();
```

4.2 FileChooser mit Filter für Text Dateien und anschließendem Auslesen

[ORA02]

5. Management

5.1. Matthias Mischek

Arbeitsbereich	Zeitlicher Aufwand (in h)
Java Programm	6
Protokoll	2
Gesamtanzahl	**8**

5.2. Benedikt Berger

Arbeitsbereich	Zeitlicher Aufwand (in h)
Java Programm	6
Protokoll	2
Gesamtanzahl	**8**

5.3. Versionierung

Wir haben GitHub für die Versionierung dieser Aufgabe verwendet.

Link zu dem Repository: https://github.com/mmischek/Prepared-Statements.git

6. Quellenverzeichnis

[BOR16] MICHAEL BORKO (TGM MOODLE) [ONLINE] AVAILABLE AT:
HTTPS://ELEARNING.TGM.AC.AT/MOD/ASSIGN/VIEW.PHP?
ID=47181&ACTION=VIEW
[ABGERUFEN AM 6. MAI 2016]

[ORA01] JAVA TUTORIAL JDBC [ONLINE] AVAILABLE AT:
HTTPS://DOCS.ORACLE.COM/JAVASE/TUTORIAL/JDBC/BASICS/
PREPARED.HTML
[ABGERUFEN AM 6. MAI 2016]

[ORA02] FILECHOOSER (JAVAFX 2.2) [ONLINE] AVAILABLE AT:
HTTP://DOCS.ORACLE.COM/JAVAFX/2/API/JAVAFX/STAGE/
FILECHOOSER.HTML
[ABGERUFEN AM 6. MAI 2016]