Übungszettel – Datenpersistenz

Name: Michael Bogensberger

Datum: 02.02.2022

Inhalt

[2 Entwicklungsumgebung einrichten 3](#_Toc95301149)

[3 Projekt anlegen 3](#_Toc95301150)

[3.1 MySql Connector Dependency hinzufügen 3](#_Toc95301151)

[4 Verbindung zu Datenbank herstellen 3](#_Toc95301152)

[5 Einfaches CRUD Beispiel mit JDBC 4](#_Toc95301153)

[6 CRUD Beispiel erweitern 4](#_Toc95301154)

[7 JDBC-Zugriff mit dem DAO Entwurfsmuster 4](#_Toc95301155)

[7.1 DAO – Data Access Object 4](#_Toc95301156)

[7.2 Singleton Pattern 6](#_Toc95301157)

# Entwicklungsumgebung einrichten

Zunächst müssen wir die Entwicklungsumgebung einrichten. Dafür installieren wir uns zunächst XAMPP. Der Name "XAMPP" ist eine Abkürzung für Apache, MySQL, Perl und PHP. Das "X" am Anfang bezieht sich darauf, dass das Programm auf verschiedenen Betriebssystemen wie Windows, Linux oder Max OS X läuft. Gegebenen falls muss man in XAMPP Ports ändern. Falls man zum Beispiel die MySql Workbench installiert hat, kann es sein das der Port 3306 bereits belegt ist. Hierfür empfiehlt sich auf den Port 3307 zu wechseln. Danach kann man im Browser die URL: localhost aufrufen. Von dort an kann man in phpMyAdmin einsteigen.

# Projekt anlegen

Als nächsten legen wir in IntelliJ ein neues Maven Projekt an. Dafür wählen wir Maven aus und verwenden dieses Mal keinen Archetype.

## MySql Connector Dependency hinzufügen

Nun fügen wir die MySql Connector Dependency hinzu. Diese wird benötigt, um mit der Datenbank kommunizieren zu können. Dazu fügen wir folgendes in der pom.xml hinzu.

|  |
| --- |
| <dependencies>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.27</version>  </dependency> </dependencies> |

Table MySql Connector Dependency

# Verbindung zu Datenbank herstellen

In folgendem Codeabschnitt wird dargestellt wie man mithilfe der MySql Connector Dependency eine Verbindung zur Datenbank herstellen kann.

|  |
| --- |
| String connectionUrl = "jdbc:mysql://localhost:3306/jdbcdemo"**;** String username = "root"**;** String pw = ""**;** try(Connection conn = DriverManager.*getConnection*(connectionUrl**,** username**,** pw)) {  System.*out*.println("DB Verbindung ok")**;** } catch (SQLException e) {  System.*out*.println("DB Verbindung fehlgeschlagen " + e.getMessage())**;** } |

Table Datenbankverbindung herstellen

# Einfaches CRUD Beispiel mit JDBC

Nun war es die Aufgabe mithilfe der Videos ein einfaches CRUD Programm mithilfe von JDBC zu gestalten. Dabei geht es im wesentlichen um das auslesen von Studenten, dem einfügen von Studenten sowie dem löschen von Studenten. Der Code zum Beispiel ist hier zu finden: [jdbcCrudExample](https://github.com/MichaelBogensberger/DatenpersistenzFSE).

Hier ist der Aufbau der Student Tabelle zu sehen:

Table Student Table

# CRUD Beispiel erweitern

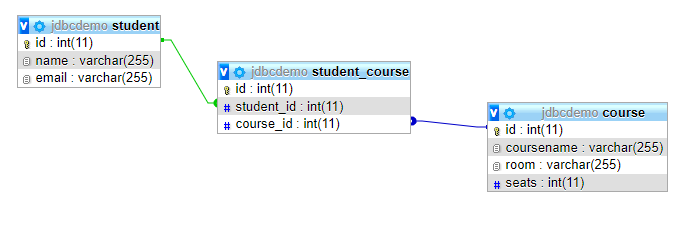
Als nächstes sollten wir das vorherige Beispiel um eine weitere Tabelle ergänzen und dieses in Java implementieren. Dazu habe ich der Datenbank eine Kurs-Tabelle hinzugefügt. Jeder Student hat mehrere Kurse. Jeder Kurs hat zudem mehrere Studenten. Der Code ist im Gleichen GitHub Repository zu finden wie in der vorherigen Aufgabe.

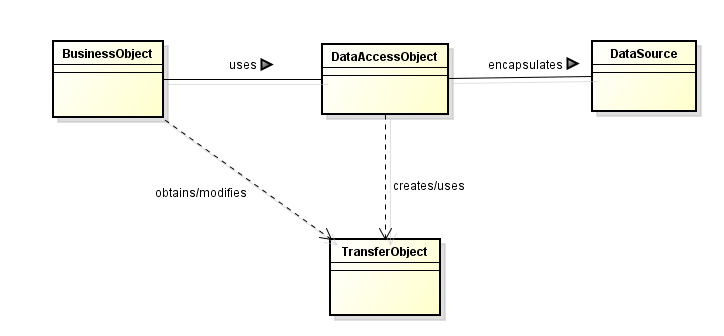
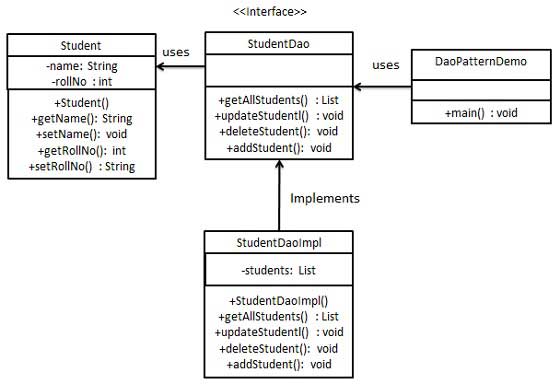
Table Student - Course - Modell

# JDBC-Zugriff mit dem DAO Entwurfsmuster

Die nächste Aufgabe war es, sich mithilfe der Videos sich mit dem DAO Entwurfsmuster vertraut zu machen.

## DAO – Data Access Object

DAO ist ein Entwurfsmuster das einem ermöglicht den Zugriff auf Daten so zu kapseln, das jene Datenquelle relativ einfach getauscht werden kann. Dadurch wird die Programmlogik von technischen Details der Datenspeicherung befreit. Man will also nicht den bestehenden Code angreifen müssen, um Funktionalität hinzufügen zu können.



## Singleton Pattern

Um die MySql Connection nur einmal aufbauen zu können verwenden wir das Singleton Pattern. Beim Singleton Pattern handelt es sich in Java um genau eine Klasse. Diese darf nur ein einziges Mal istanziert werden. Während der Programmlaufzeit existiert nur ein einziges Objekt. Das Singleton Pattern wurde zudem schon einmal von mit beim Übungszettel Mikroarchitektur beschrieben.

Dabei erstellen wir die MySqlDatabaseConnection Klasse. Diese liefert die Connection zurück. Hier ist es eben wichtig das nur eine Connection existiert. Dabei setzen wir den Konstruktor au private. Somit kann dieser nicht aufgerufen werden. Danach checken wir ob die Connection bereits existiert. Existiert diese nicht, wird eine neue Connection erstellt und zurückgegeben.

|  |
| --- |
| *public class* MySqlDatabaseConnection {  *private static Connection con* = *null*;    *private* MySqlDatabaseConnection() {  }  *public static Connection* getConnection(String url, String user, String pwd) *throws* ClassNotFoundException, SQLException {  *if*(*con* != *null*) {  *return con*;  } *else* {  Class.*forName*("com.mysql.cj.jdbc.Driver");  *return con* = DriverManager.*getConnection*(url, user, pwd);  }  } } |

Table MySqlDatabseConnection Klasse

|  |
| --- |
| *try* {  *Connection* myConnection = MySqlDatabaseConnection.*getConnection*("jdbc:mysql://localhost:3307/kurssystem", "root", "");  System.*out*.println("Verbindung aufgebaut"); } *catch* (ClassNotFoundException e) {  e.printStackTrace(); } *catch* (SQLException e) {  e.printStackTrace(); } |

Table Verbindungserstellung in der Main Methode