Übungszettel – Datenpersistenz

Name: Michael Bogensberger

Datum: 02.02.2022

Inhalt

[2 Entwicklungsumgebung einrichten 3](#_Toc96510519)

[3 Projekt anlegen 3](#_Toc96510520)

[3.1 MySql Connector Dependency hinzufügen 3](#_Toc96510521)

[4 Verbindung zu Datenbank herstellen 3](#_Toc96510522)

[5 Einfaches CRUD Beispiel mit JDBC 4](#_Toc96510523)

[5.1 CRUD Beispiel erweitern 4](#_Toc96510524)

[6 JDBC-Zugriff mit dem DAO Entwurfsmuster 5](#_Toc96510525)

[6.1 DAO – Data Access Object 5](#_Toc96510526)

[6.2 Singleton Pattern 6](#_Toc96510527)

[6.3 Cli Klasse 7](#_Toc96510528)

[6.4 Domain Package 8](#_Toc96510529)

[6.5 Base Repository 8](#_Toc96510530)

[6.6 MyCourseRepository Repository 9](#_Toc96510531)

[6.7 MySqlCourseRepository Klasse 9](#_Toc96510532)

[7 ResultSet.next() Hinweis 12](#_Toc96510533)

[8 Abbildungsverzeichnis 13](#_Toc96510534)

# 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.

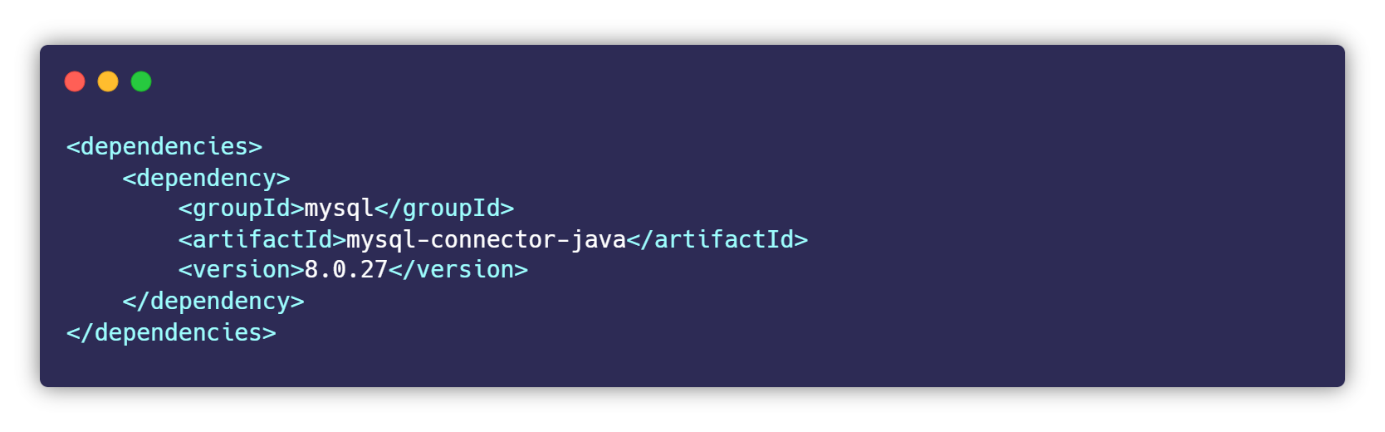


Abbildung MySql Connector Dependency

# Verbindung zu Datenbank herstellen

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

Abbildung 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 3 Student Table

## CRUD Beispiel erweitern

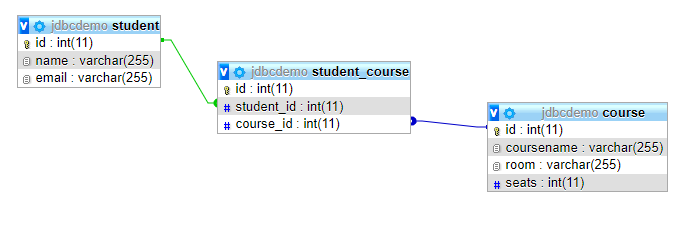
Als nächstes sollten wir das vorherige Beispiel um eine weitere Tabelle ergänzen und dieses in Java implementieren. Dazu habe ich der Datenbank eine Kurs-Tabelle hinzugefügt. Jeder Student hat mehrere Kurse. Jeder Kurs hat zudem mehrere Studenten. Der Code ist im Gleichen GitHub Repository zu finden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

Table 4 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. DAO wird also zwischen der Datenbank und dem Code geschalten. Dadurch wird die Kopplung auf ein Minimum heruntergefahren. Wenn sich der JDBC Treiber zum Beispiel ändert, ist der grundsätzliche Code unabhängig. Das entkoppeln ist das zentrale des DAO Design Patterns.

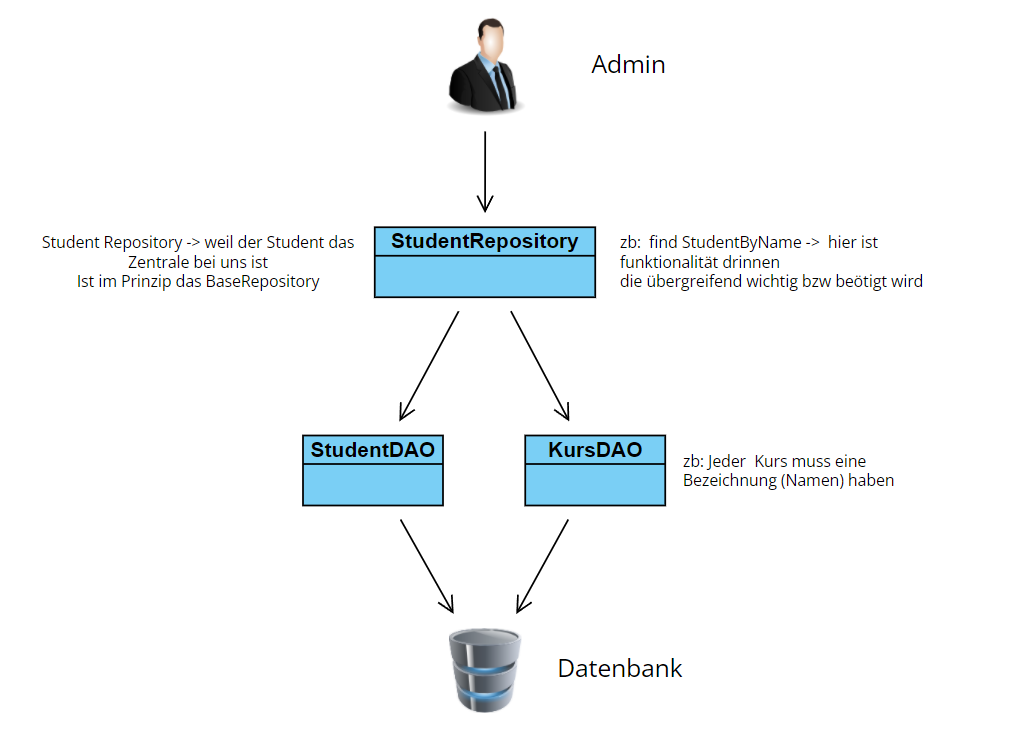


Abbildung Kurssystem DAO

Wie wir sehen können bringt uns also DAO den großen Vorteil der geringen Kopplung. Dies ist auch nochmal in folgender Grafik veranschaulicht.

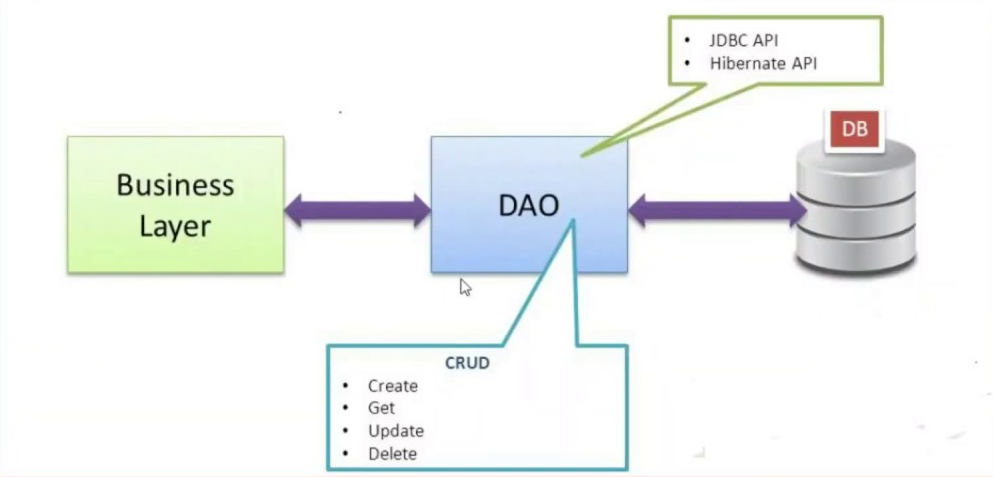


Abbildung DAO Design Pattern / Kopplung

## 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.



Abbildung MySqlDatabaseConnection Klasse

Abbildung Verbindungserstellung in der Main Methode

## Cli Klasse

Für die Ausgabe erstellen wir uns eine eigene Cli Klasse. Dort erstellen wir uns im Konstruktor einen Scanner und ein Switch Case mit dem wir durch die verschiedenen Auswahlmöglichkeiten iterieren.

Abbildung Cli Klasse

## Domain Package

Nun erstellen wir uns ein Domain Package und darin die Entitäten als Klassen. Zunächst legen wir eine Course Klasse und eine Course Type Klasse an. Die Course Type Klasse ist ein Enum und enthält die verschiedenen Course Typen. Danach erstellen wir eine BaseEntity Klasse. Jene ist dafür da, die ID nicht immer händisch eintragen zu müssen. Sie kümmert sich also um die Logik der IDs. In der Course Klasse haben wir die jeweiligen Getter und Setter und zwei Konstruktoren. Einen bei dem die ID mit übergeben wird und gesetzt wird, und einen wo keine ID mit übergeben wird. In diesem Fall wird die ID auf NULL gesetzt. **WIESO**

Nun schaut unser Package folgendermaßen aus:

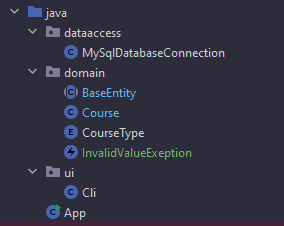


Abbildung 8 Package Struktur

## Base Repository

Nun erstellen wir uns ein BaseRepository. Dieses kann ich nun für jede Art von Entität verwenden. Dabei werden Generics verwendet. Dabei gibt es einmal T für den Typen und I für die ID. Folgende Methoden müssen implementiert werden.

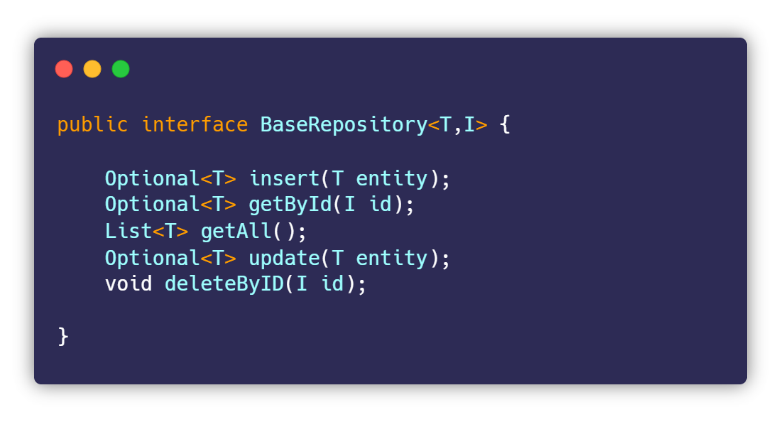


Abbildung 9 BaseRepository

## MyCourseRepository Repository

Nun erstellen wir ein CourseRepository. Dieses erbt das BaseRepository. Dabei wird dieses mit Course und Long getypt. Course als Entitytyp und Long als Schlüsseltyp. Der Vorteil ist, dass ich das BaseRepository immer verwenden kann und jetzt zb im CourseRepository zusätzlich Funktionen spezifizieren.



Abbildung 10 MyCourseRepository

## MySqlCourseRepository Klasse

Als nächstes erstellen wir eine MySqlCourseRepository Klasse. Hier wird jetzt klar gemacht welche Technologie (MySql) verwendet wird. Diese Klasse implementiert nun die MyCourseRepository Klasse. Nun müssen wir also alle Methoden der beiden obigen Repositories implementieren.

Dazu erstellen wir uns zunächst eine Connection. Da wir ja die MySqlDatabaseConnection Klasse haben können wir diese dafür verwenden.

Abbildung Database Connection (MySqlCourseRepository)

Nun fangen wir an die getAll() Methode mit Funktionalität zu befüllen. In der while Schleife befüllen wir nun die ArrayList mit den jeweiligen Werten. Hier sollte man genau auf die Labels achten! Nun haben wir ja eine Enumeration für den Kurstypen (CourseType). Wir bekommen jedoch einen String aus der DB zurück. Deshalb müssen wir hier den String noch in einen CourseType konvertieren. Diese sieht nun wie folgt aus:

Abbildung getAll Methode (MySqlCourseRepository)

Nun gehen wir in die Cli Klasse und fügen im Konstruktor ein MyCourseRepository hinzu. Danach fügen wir im Switch Case im zweiten Case die Methode showAllCourses hinzu. Diese müssen wir natürlich noch implementieren.

Abbildung Cli Klasse (repo)

## showAllCourses Methode

Nun können wir die showAllCourses Methode implementieren. Dazu erstellen wir zunächst eine List vom Typ Course. Danach befüllen wir die Liste über das Repository mit getAll(). Nun checken wir ob die liste leer ist. Ist sie es nicht, gehen wir durch die jeweiligen Kurse in der Liste durch und geben diese aus. Dazu haben wir ja auch die toString Methode implementiert. Jene wird nun im „sout“ verwendet. Danach umgeben wir das ganze noch mit einem try catch und sagen was passiert wenn Fehler auftreten.

Abbildung showAllCourses Methode

Schließlich können wir in die App Klasse bzw. Main Methode wechseln. Hier erstellen wir ein Cli Objekt. Das wichtige ist nun, dass wir dem Cli Objekt ein MySqlCourseRepository mitgeben.

Abbildung App Klasse

Das Zusammenspiel mit den Repositories wird in folgender Grafik nochmal genauer dargestellt. Hier ist gut zu sehen welche Klasse welche Repositories verwendet. Zurzeit verwendet die Cli nur das MyCourseRepository. Das kann sich jedoch mit zusätzlicher Funktionalität ändern.

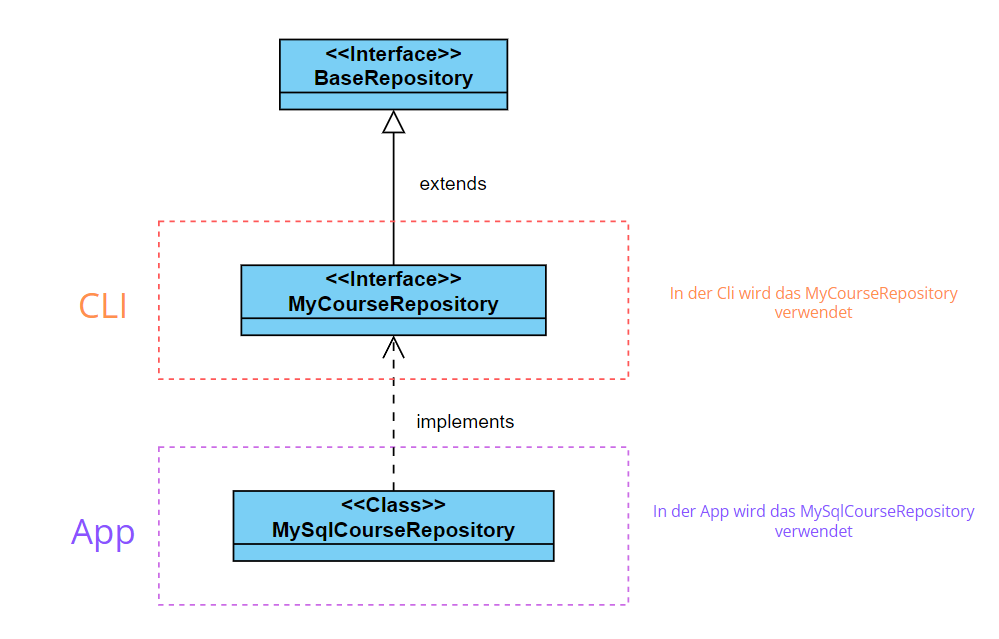
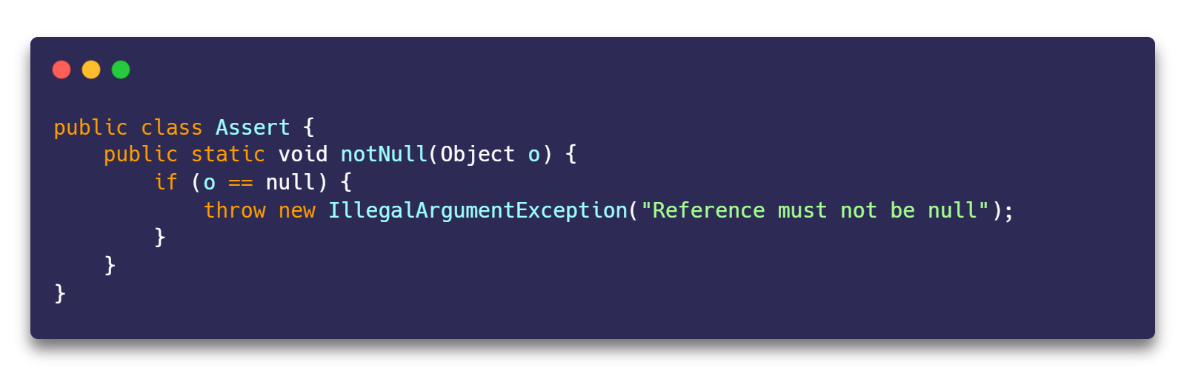


Abbildung Grafik - Repositories

## Assert Klasse

Nun erstellen wir uns ein util Package. Darin erstellen wir uns eine Assert Klasse. Diese dient dazu zu checken ob ein Objekt null ist oder nicht.



## Kurs durch ID bekommen (getById Methode)

Nun programmieren wir die Funktionalität der getById Methode.



# ResultSet.next() Hinweis

Wenn ich mehrere Rows (Zeilen) zurück bekomme ist es klar das ich mit .next() durch die jeweiligen Zeilen springen muss. Wenn ich jedoch nur eine Zeile zurückbekomme muss ich auch einmal .next() ausrufen. Deshalb ja nicht den Befehl .next() vergessen!

# Abbildungsverzeichnis

[Abbildung 1 MySql Connector Dependency 3](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508439)

[Abbildung 2 Datenbankverbindung herstellen 3](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508440)

[Abbildung 3 Kurssystem DAO 5](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508441)

[Abbildung 4 DAO Design Pattern / Kopplung 5](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508442)

[Abbildung 5 MySqlDatabaseConnection Klasse 6](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508443)

[Abbildung 6 Verbindungserstellung in der Main Methode 6](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508444)

[Abbildung 7 Cli Klasse 7](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508445)

[Abbildung 8 Package Struktur 8](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508446)

[Abbildung 9 BaseRepository 8](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508447)

[Abbildung 10 MyCourseRepository 9](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508448)

[Abbildung 11 Database Connection (MySqlCourseRepository) 9](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508449)

[Abbildung 12 getAll Methode (MySqlCourseRepository) 9](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508450)

[Abbildung 13 Cli Klasse (repo) 10](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508451)

[Abbildung 14 showAllCourses Methode 10](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508452)

[Abbildung 15 App Klasse 10](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508453)

[Abbildung 16 Grafik - Repositories 11](file:///D:\dev\FSE\DatenpersistenzFSE\Datenpersistenz.docx#_Toc96508454)