Ausarbeitung

Programmieren in Clojure

ClojureMySQLEditor

von

Niklas Simonis

Dominik Eller

März 2014

Inhaltsverzeichnis

1 Einleitung

Im Kurs "Programmieren in Clojure" an der Technischen Hochschule Mittelhessen (THM) im Wintersemester 2013/14 wurde den Studenten die Programmiersprache Clojure in der Theorie vorgestellt. Durch wöchentliche Übungen konnten zudem praktische Erfahrungen gesammelt werden. Im Rahmen der Vorlesung wurden die Studenten in Teams eingeteilt, um das gelehrte Wissen praktisch anwenden zu können. Jedes Team hat ein bestimmtes Thema zur Umsetzung zugewiesen bekommen oder konnte sich ein eigenes aussuchen. Das Thema für unser aus zwei Studenten bestehendes Team ist die Implementierung einer grafischen Benutzeroberfläche zur Bedienung von Datenbanken. Die Benotung für das Fach "Programmieren in Clojure" setzt sich für jedes Team aus dessen Implementierung der jeweiligen Aufgabe und der dazu gehörigen Ausarbeitung zusammen.

2 Aufgabenbeschreibung

In diesem Kapitel werden die Funktionalitäten und die eingesetzten Technologien vorgestellt. Zudem wird die Zielgruppe definiert.

2.1 Welche Funktionalitäten sollen zur Verfügung stehen?

Der Anwender soll die Möglichkeit haben sich mit einer Datenbank seiner Wahl zu verbinden. Anschließend kann er über ein Auswahlfeld eine Tabelle der Datenbank auswählen und bekommt deren Inhalt in einer Tabelle angezeigt. Für die jeweils ausgewählte Tabelle sollen dem Anwender Funktionen wie Einfügen eines neuen Datensatzes, Bearbeiten und Löschen eines bestehenden Datensatzes zur Verfügung stehen. Zudem besteht mittels Eingabekonsole eine weitere Möglichkeit auf die Datenbank zuzugreifen. Dort kann der Anwender direkt SQL - Befehle ausführen. Ein weiteres Feature für den Anwender soll die Export-Funktion bieten. Mittels dieser Funktion soll der Anwender die jeweils ausgewählte Tabelle in eine SQL-Datei exportieren können.

2.2 Welche Technologien sollen verwendet werden?

Die folgende Auflistung zeigt, welche Technologien in diesem Projekt verwendet wurden.

- Clojure
- Java
- Java JDBC-Treiber
- Java MySQL-Connector

2.3 Welche Zielgruppen sollen erreicht werden?

Die Zielgruppe für diese Anwendung sollen Privatanwender sein, die schnell und unkompliziert auf ihre Datenbank zugreifen möchten, um Inhalte anzusehen, zu bearbeiten, zu löschen oder neu einzufügen. Die Anwendung ist also speziell auf die Bedürfnisse des Endkunden optimiert und administrative Funktionen wurden bewusst ausgelassen. Im Vergleich zu anderen Editoren, wie zum Beispiel phpMyAdmin, wird der Endkunde nicht mit diesen Zusatz Features überfordert und kann schnell und unkompliziert mit seiner Datenbank arbeiten.

3 Konzeptbeschreibung

In diesem Kapitel werden die Ergebnisse der Anforderungsanalyse dargestellt. Zudem wird darauf eingegangen, warum Java statt Seesaw eingesetzt wird. Des weiteren wird auf das Entwurfsmuster MVC (model, view und controller) eingegangen. Das Klassendiagramm wird anhand eines FMC-Diagramms verdeutlicht und das Projekt wird dokumentiert.

3.1 Anforderungsanalyse

Die folgende Auflistung zeigt alle Anforderungen an die zu erstellende Software:

- Verbindung mit einer Datenbank herstellen.
- Alle Tabellen einer Datenbank sollen auswählbar sein.
- Inhalte einer Tabelle sollen dargestellt werden.
- Hinzufügen eines neuen Datensatzes in einer Tabelle.
- Bearbeiten eines Datensatzes in einer Tabelle.
- Löschen eines Datensatzes in einer Tabelle.
- Ausführen von SQL-Kommandos, hierbei sollen keine grafischen Anzeigen manipuliert werden.
- Exportieren der Tabellen soll möglich sein.
- Erweiterbarkeit für andere Datenbanken in späteren Versionen.

3.2 Erweiterbarkeit der Datenbanktypen

Uns war es wichtig, die Erweiterbarkeit für spätere Versionen zu gewährleisten. Bisher arbeitet die aktuelle Version nur mit MySQL Datenbanken. Hier sollten später auch andere Treiber, wie z.B. PostgreSQL uvm., unterstüzt werden. Daher wurde der Quellcode so entwickelt, dass die Erweiterbarkeit gegeben ist und mit weniger Aufwand realisierbar ist.

3.3 Java statt Seesaw

Seesaw ist ein Interface in Clojure zum Erstellen von grafischen Benutzeroberflächen, welches auf Java Swing aufsetzt. Wir haben uns gegen Seesaw entschieden und direkt Java als

GUI Builder verwendet, weil die individuelle Formatierung und Ausrichtung der Ansicht mittels Seesaw erfordert hat, dass man sowieso direkt auf die Java Swing Objekte zugreifen muss. Aus diesem Grund war es übersichtlicher und einfacher direkt Java zu verwenden.

3.4 Design

Die folgenden Abbildungen zeigen einige grafische Mockups, wie die Software aussehen soll.

3.4.1 Login

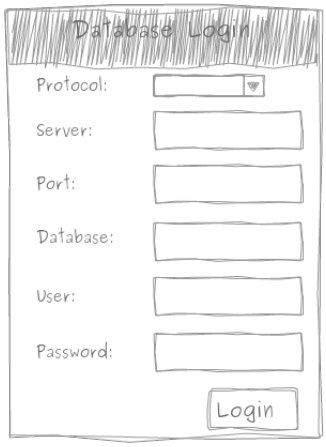

Abbildung 3.1: Login-Fenster mit allen Informationen

Das Login Fenster soll die Möglichkeiten bieten, einen Treiber für die zu verwendende Datenbank auszuwählen. Ebenso soll der Benutzer die Möglichkeit haben, einen Datenbankserver inklusive dessen Port anzugeben. Der Benutzer soll den Namen der Datenbank und seine Login Informationen eintragen können.

3.4.2 Editor

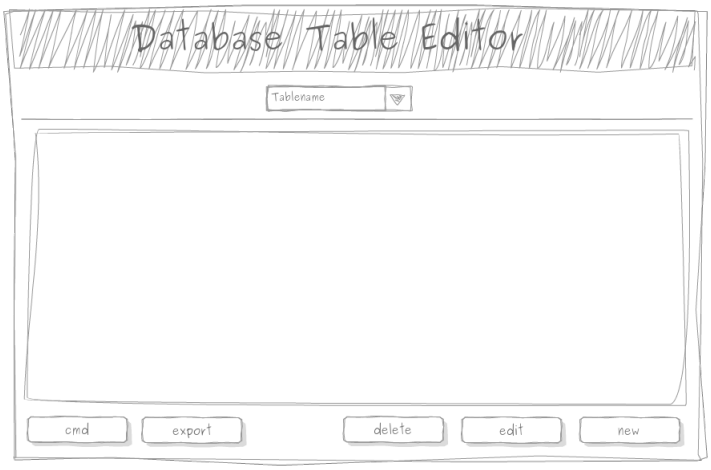

Abbildung 3.2: Hauptfenster des Editors

Der eigentliche Editor soll dem Anwender die Möglichkeit geben, möglichst einfach zwischen den Tabellen einer Datenbank mittels Dropdown hin und her zu springen. Unter dem Dropdown wird ihm dabei stets der aktuelle Inhalt der Tabellen angezeigt. Der Anwender soll die Möglichkeit haben, ein SQL-Command auszuführen, die Tabelle zu exportieren, einen Eintrag zu löschen, zu bearbeiten und einen neuen Eintrag in die Tabelle hinzuzufügen.

3.4.3 Editieren

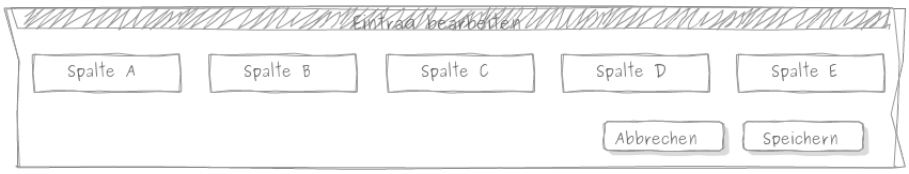

Abbildung 3.3: Editieren-Fenster eines Eintrages

Der Anwender bekommt ein Popup mit der ausgewählten Spalte angezeigt, sobald er auf *edit* geklickt hat. Dort kann er Einträge manipulieren und in der Datenbank speichern.

3.4.4 SQL Command

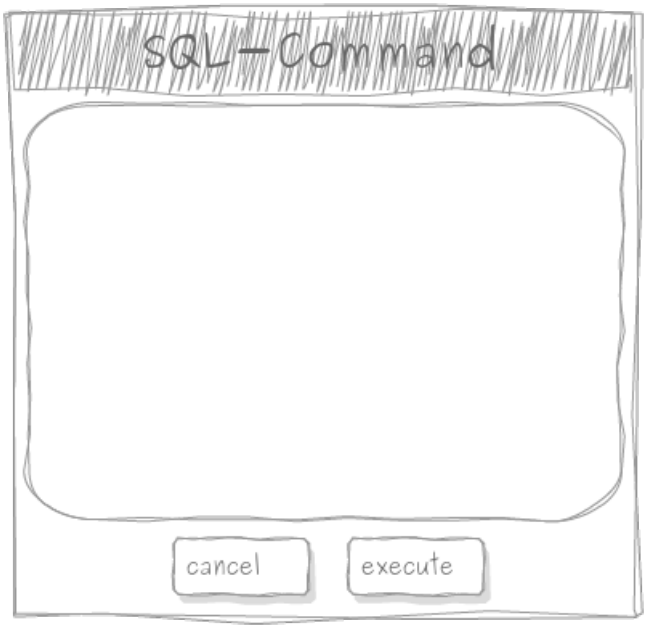

Abbildung 3.4: SQL-Command-Fenster für die Datenbank

Wenn der Anwender auf den Button “cmd” drückt, soll sich ein Fenster öffnen, in welchem der Anwender die Möglichkeit hat, SQL-Statements einzugeben und auszuführen. Die Ergebnisse des Ausführens werden allerdings nicht grafisch dargestellt.

3.4.5 Error-Frame

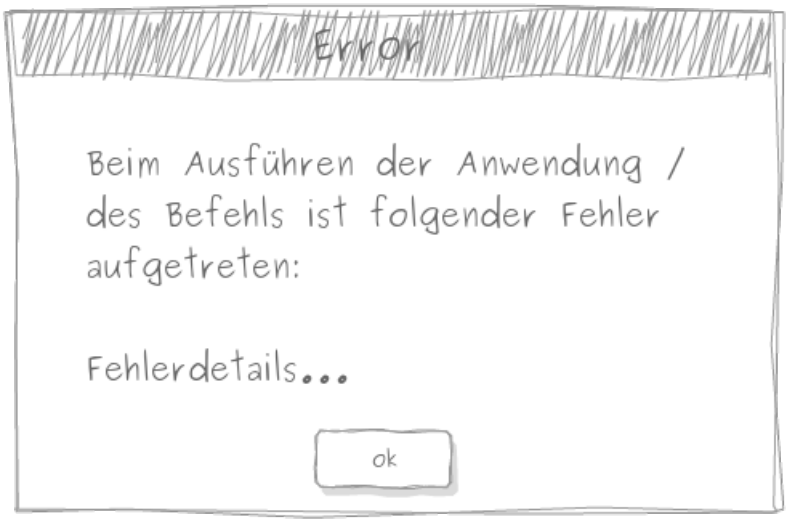

Abbildung 3.5: Error-Fenster für die Anwendung

Tritt in der Anwendung ein Fehler auf, oder passiert etwas unerwartetes, wird der Anwender über ein Popup über den Fehler und dessen Details informiert.

3.5 Modularität

Das Design der Anwendung ist an das MVC (Model, View, Controller) Entwurfsmuster angelehnt. Der Quellcode ist auf 4 Dateien aufgeteilt. Dabei je eine Datei für Model (*model.clj*), View (*view.clj*) und Controller (*controller.clj*) und einer Haupt- bzw. Startdatei (*core.clj*). Wird der Umfang der Anwendung größer, macht es sicherlich Sinn, den Quellcode noch strikter zu kapseln und für jede View eine eigene Datei anzulegen.

3.6 FMC-Diagramm

Die folgende Abbildung zeigt ein FMC-Diagramm, welches die Software beschreibt. Der Database Table Editor hat die einzelnen Funktionen insert, update, delete, sql cmd, login und export. Diese Optionen können vom Anwender ausgeführt werden. Der Database Table Editor wiederum manipuliert dann die jeweilige Datenbank.

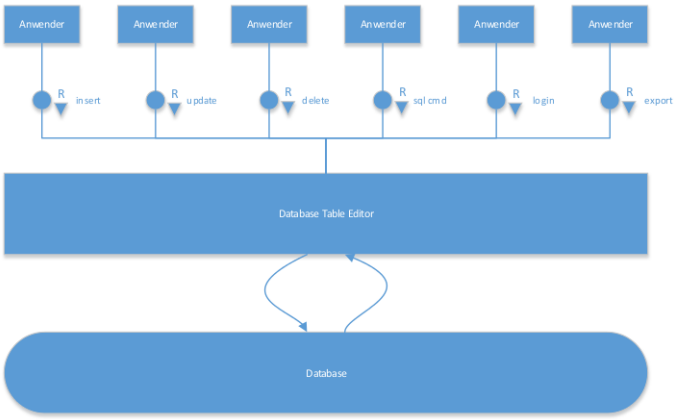

Abbildung 3.6: FMC-Diagramm der Anwendung

3.7 Klassendiagramm

Die folgende Abbildung zeigt ein Klassendiagramm. Es sind alle globalen Variablen und Funktionen in den jeweiligen Dateien aufgeführt.

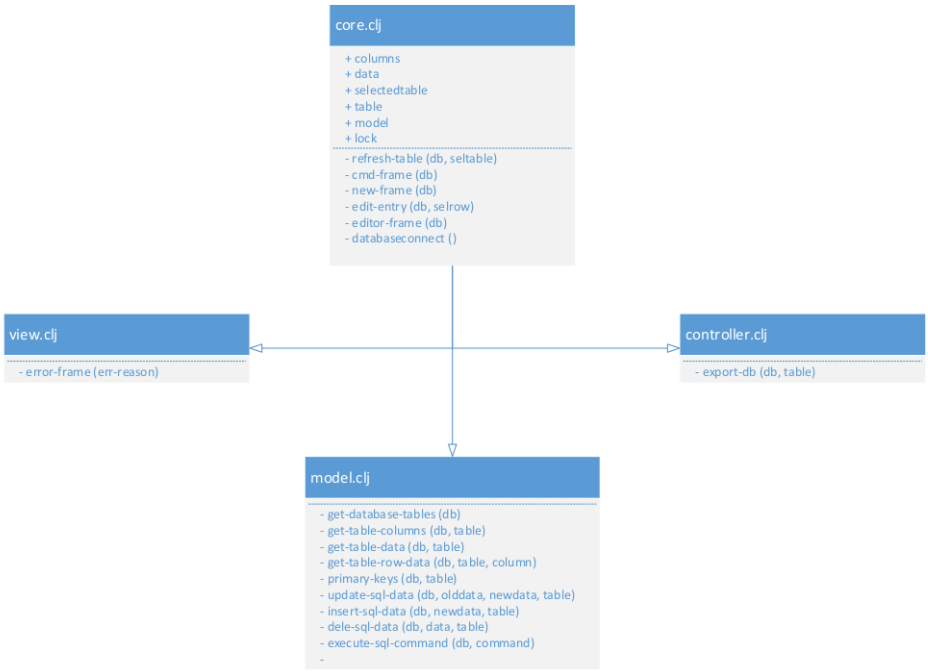

Abbildung 3.7: Klassendiagramm der Anwendung

3.8 Dokumentation

Als Dokumentation stehen neben der Ausarbeitung auch eine beigelegte Textdatei (Read-Me.md) als Hilfestellung zur Verfügung. Diese enthält neben der Installation des Projektes auch allgemeine Informationen.

3.8.1 Bedienungsanleitung

Nachdem man das Programm gestartet hat, wird man in einem Fenster aufgefordert, Login Informationen für die Datenbankverbindung einzutragen.

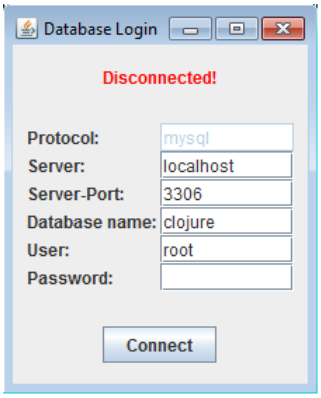

Abbildung 3.8: Database Login Fenster

Als Protokoll ist fest das MySQL-Protokoll vorausgewählt. Das Protokoll kann in dieser Version nicht verändert werden. Bei Server trägt man die URL oder IP-Adresse des Datenbankservers ein. Zum Beispiel “localhost” oder eine IP Adresse. Der Standard Port für MySQL-Datenbanken ist “3306”. Dieser ist bereits eingestellt und muss in der Regel nicht verändert werden. Als Datenbank Name gibt man den Namen der Datenbank ein, mit der man sich verbinden möchte. In diesem Beispiel heißt die Datenbank “clojure”. Bei “User” und “Password” trägt man die Login Informationen des MySQL-Benutzers ein. Sind alle Eingaben vollständig, drückt man auf “Connect” und das Programm verbindet sich mit dem Server. Wenn die Informationen falsch, sind erscheint eine Fehlermeldung. War der Login erfolgreich, wird das Fenster automatisch geschlossen und ein neues Fenster erscheint. Siehe folgende Abbildung.

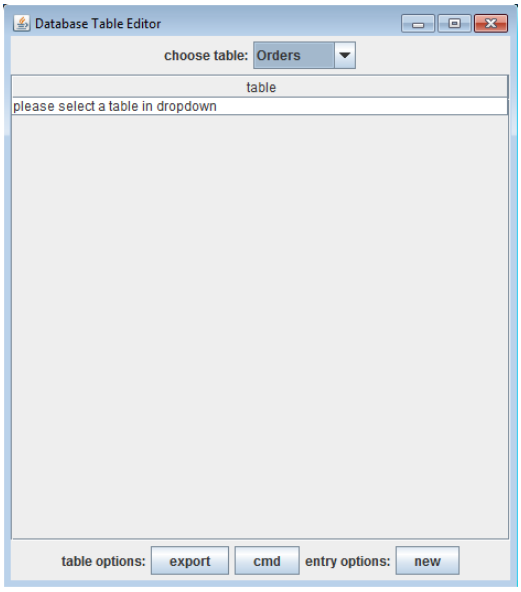

Abbildung 3.9: Editor Hauptfenster

In diesem Fenster kann man alle Tabellen einer Datenbank auswählen. Wählt man eine Tabelle aus, mit der man arbeiten möchte, wird diese grafisch in einer Tabelle angezeigt. In der folgenden Abbildung wurde beispielsweise die Tabelle “books” aus der Datenbank “clojure” ausgewählt.

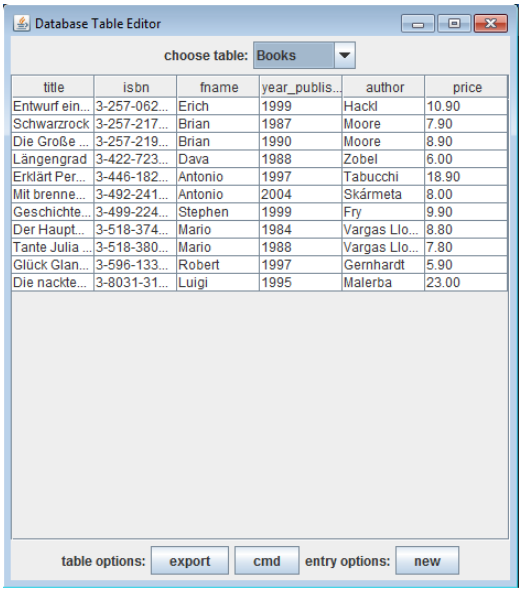

Abbildung 3.10: Editor Hauptfenster mit Tabelleninhalt

Man hat nun die Möglichkeit, Inhalte der Tabelle zu bearbeiten. Dazu klickt man auf eine Zeile der Tabelle und es öffnet sich automatisch ein Popup mit dem Inhalt des ausgewählten Datensatzes.

Abbildung 3.11: Edit Ansicht einer Zeile

Diesen Inhalt kann man direkt bearbeiten und anschließend speichern oder durch einen Klick auf den Button “delete” aus der Datenbank löschen. Wird das Popup geschlossen, bekommt man die bereits aktualisierte Tabelle angezeigt. Wenn man einen neuen Datensatz zur Tabelle hinzufügen möchte, klickt man auf den Button “New” und es öffnet sich ein Eingabeformular.

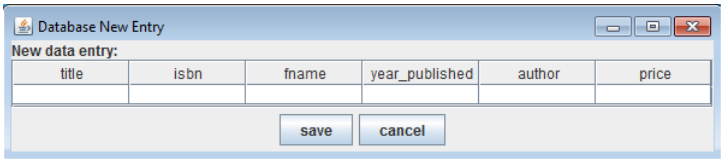

Abbildung 3.12: New Ansicht für neuen Eintrag

In diesem Eingabeformular gibt man nun die gewünschten Informationen ein und drückt

anschließend auf "save". Der Eintrag wird automatisch hinzugefügt und die Tabelle wird im Hauptfenster aktualisiert. Möchte man seine Eingaben verwerfen, kann man einfach den "cancel" Button drücken.

Eine weitere Möglichkeit auf die Datenbank zuzugreifen besteht mittels Eingabekonsole, welche durch einen Klick auf den Button "cmd" geöffnet werden kann. In dem sich öffnenden Popup hat der Anwender die Möglichkeit, direkt SQL-Befehle einzugeben und auszuführen.

Ist der SQL - Befehl eingegeben, klickt man auf "execute" und der eingegebene Befehl wird in der Datenbank ausgeführt. Es wird kein Ergebnis grafisch dargestellt. Die Operation wird jedoch auf der Datenbank ausgeführt. Es macht also wenig Sinn, als SQL - Befehl einen SELECT oder JOIN zu verwenden, da das Ergebnis nicht angezeigt wird.

Vielmehr sind Befehle wie INSERT, UPDATE oder DELETE gedacht. Aktualisiert man später von Hand die Tabelle, sieht man auch die Ergebnisse des Befehls.

Drückt man im Hauptfenster auf den Button "export", wird die ausgewählte Tabelle in eine Datei mit der Endung .sql exportiert. Dazu öffnet sich ein Dialog. Hier muss ein Speicherort ausgewählt und ein Dateiname eingetragen werden.

3.8.2 Installationsanleitung

Die Installationsanleitung beschreibt, wie man die Anwendung in Eclipse ausführen und starten kann. Es ist auch möglich, eine einfache ausführbare Datei mittels Java zu starten.

Importieren der Quelldateien

Das Projekt aus Github (URL: https://github.com/NiklasLM/clj-db-project) in ihr Eclipse importieren (Git Plugin muss vorinstalliert sein).

Projekt anlegen

Neues Clojure Projekt in Eclipse mit dem Namen "ClojureMySQLEditor" anlegen (Clojure Integration muss in Eclipse verfügbar sein).

Quelldateien kopieren

Kopieren Sie alle Quelldateien in das neue angelegte Clojure Projekt. Das Projekt sollte dann wie in folgender Abbildung aussehen.

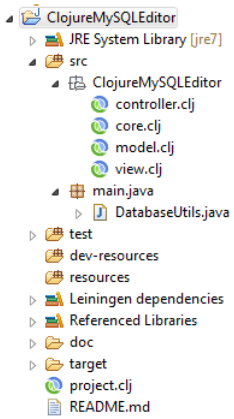

Abbildung 3.13: Clojure Projekt im Eclipse

Build Path des Projektes anpassen

Als nächstes muss der Build Path angepasst werden. Dazu machen Sie einen Linksklick auf das "ClojureMySQLEditor" Projekt und danach einen Rechtsklick. Dann klicken Sie ganz unten auf Properties. Daraufhin öffnet sich ein neues Fenster. Klicken Sie links im Menü auf Java Build Path. Nun klicken Sie rechts auf Add External JARs. Die Treiber sind im Project aus Github im Unterordner "data" enthalten. Nach dem Einbinden sollte das Fenster wie in folgender Abbildung aussehen.

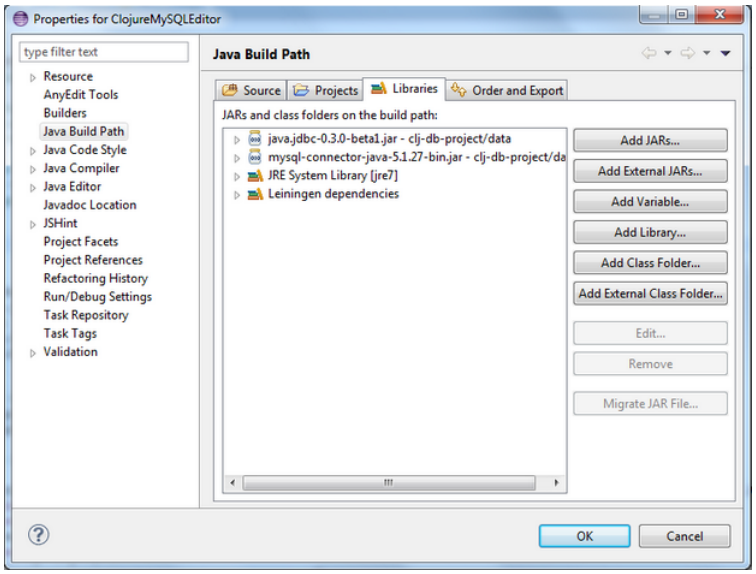

Abbildung 3.14: Build Path Einstellungen in Eclipse

Starten der Anwendung

Nun können Sie mittels Rechtsklick in der core.clj und Clojure - Load file in REPL die Anwendung starten und es erscheint das beschriebene Login Fenster.

3.8.3 Codedokumentation

Als Dokumentation wurde zu Beginn jeder Datei ein Kommentarblock mit Informationen eingefügt. Ebenso wurde zu jeder Funktion ein Kommentar Block eingefügt. Es wurden folgende Informationen zu den jeweiligen Blöcken definiert und bereitgestellt.

Kommentarblock Datei

Dokumentation zu Beginn einer Datei.

@version Versionsnummer

@package Paketname

@name Name der Datei

@author Autoren

@description Beschreibung der Datei

@link Webseite des Projektes (Github)

Kommentarblock Funktion

Dokumentation der Funktionen als Kommentar über der Funktion.

@name Funktionsname

@description Beschreibung der Funktion

@param Parameter der Funktion

@return Rückgabewert

4 Problemstellungen

Dieses Kapitel beschreibt die Problemstellungen, die während des Projektes aufgetreten sind und mit Umwegen gelöst wurden.

4.1 ActionListener für JTable

Da ein ActionListener für die JTable nicht funktioniert hat, wurde hier ein ListSelection-Listener verwendet. Dieser prüft bei jeder Änderung der JTable, ob ein "lock" gesetzt ist oder nicht. Dieses verhindert eine falsche Ausführung. Ebenso wird verhindert, dass der Listener doppelt ausgeführt wird.

4.2 Datenbank

SQL Datenbank enthält kein Attribut "id", daher mussten hier die Primary Keys der Datenbank ausgelesen werden und den Tabellen zugeordnet werden. Änderungen an einem Eintrag werden dann mittels des Primary Keys der jeweiligen Tabelle angepasst.

4.3 SQL Export

Der SQL Export stellte sich als sehr schwierig heraus. Daher wurde hier die Java-Klasse "DatabaseUtils" aus einem vorangegangenen Modul Datenbanken und Informationssysteme eingebunden. Die Klasse wurde an die Anforderungen angepasst, welche vier Funktionen enthält.

5 Fazit

Alles in Allem war es eine schöne Erfahrung, eine neue Sprache kennenzulernen. Clojure macht Java noch einfacher und eleganter, gerade im Bezug auf das Oberflächendesign. Jedoch befindet sich Clojure noch in der Entwicklung. Das wirkt sich im Besonderen auf die Dokumentation der Entwickler aus. Für Einsteiger ist es schwierig, auf Anhieb ein umfangreiches Programm ohne Fehler zu implementieren. Man hangelt sich vielmehr von einer Problemstellung zur nächsten und schreibt ein Workaround nach dem anderen. Im Netz findet man leider sehr wenig Beispiele, weil bisher wenige Personen Erfahrungen mit Clojure gesammelt haben. Wir sind jedoch der Meinung, dass man die Scheu bzw. die Angst vor etwas Neuem ablegen sollte und sich ruhig auf Clojure einlassen kann. Das Potential der Sprache ist jedenfalls sehr groß.

Abbildungsverzeichnis