m29987

Programmieren 3 - Hausarbeit 2023

Dokumentation

Aufgabe:

Die Aufgabe war es einen REST Service für Informatik Fachliteratur in Java zu erstellen. Der Service soll JSON basiert sein.

Server:

Soll mit Anfragen von mehreren Klienten umgehen können. Er besitzt eine Verbindung zu einer Datenbank die die Daten zur Fachliteratur enthält. Der Server nimmt Anfragen entgegen und kommuniziert entsprechend mit der Datenbank.

Klienten:

Sollen mit JavaFX implementiert werden. Sie können Anfragen an den Server senden und die erhaltenden Daten in POJOs umwandeln und darstellen.

Es kann einen Hauptbenutzer geben, der zusätzlich Daten in der Datenbank erstellen / bearbeiten / löschen kann.

Starten des Servers:

Um die Anwendung benutzen zu können muss zuvor mit einem Programm wie XAMPP eine Datenbank dem Programm zur Verfügung gestellt werden. Es wird das Vorgehen mit XAMPP beschrieben:

1. XAMPP Launcher starten, MySQL und Apache starten.
2. phpMyAdmin starten.
3. Unter „Datenbanken“ eine neue Datenbank mit Name: „informatik“ erstellen.
4. Auf Benutzerkonten wechseln und neuen Benutzer anlegen.
5. Name: „minf“ und Passwort „prog3“ eingeben, sowie alle Rechte verteilen.
6. Unter „Datenbank“ die Datenbank „informatik“ wählen und alle rechte vergeben
7. phpAdmin kann nun geschlossen werden.

Der Server kann über Gradle gestartet werden. Beim ersten Mal starten wird die Datenbank automatisch gefüllt.

Soll der Server nicht nur auf dem lokalen System laufen, kann unter „ServerResources.properties“ von „localhost“ auf eine IP Adresse gewechselt werden.

Starten des Klienten:

Die Klienten können über Gradle gestartet werden.

Der Klient startet immer im Login Bildschirm. Von dort kann er als normaler Benutzer oder als Hauptbenutzer das Programm betreten.

Entscheidet sich der Benutzer den Benutzerstatus zu wechseln, werden seine Einstellungen dennoch übernommen.

Wird der Klient ohne den Server gestartet, enthält er auch keine Daten. Außerdem kann man sich nicht als Hauptbenutzer anmelden.

Wenn der Server gestartet wurde, kann man sich als Hauptbenutzer einloggen. Nur ein Klient kann jeweils Hauptbenutzer werden. Dieser Klient muss sich erst wieder abmelden oder das Fenster schließen, bevor der Hauptbenutzer wieder freigegeben wird.

Voreinstellung des Host Namens im Klienten ist „localhost“, kann aber auch auf eine entsprechende Adresse gestellt werden.

Normaler Benutzer:

Der Normale Benutzer kann die Bücher in der Datenbank anfragen und darstellen.

Er erhält eine Liste der Bücher und kann per Doppelklick mehr Informationen zu einem Buch in einer neuen Ansicht ansehen.

Die Listengröße (Seitengröße) kann vom Benutzer angepasst werden. So können mehr oder weniger Bücher auf einer Seite angezeigt werden. Der Benutzer kann durch die Seiten wechseln.

Der Benutzer kann die Sortier-Richtung einstellen. Es wird nach Titel sortiert.

Der Benutzer kann die Bücher nach Kategorien filtern. Ebenso kann er auch eine Zeichenfolge eingeben. Titel, die die Zeichenfolge enthalten werden markiert und gefiltert.

Filter werden erst nach Bestätigung durch den Filter-Knopf angewandt.

Bestehende Filter können durch den Zurücksetzen-Knopf zurückgesetzt werden.

Mit dem Return-Knopf kann der Benutzer zurück ins Login Menü.

Hauptbenutzer:

Der Hauptbenutzer kann alles, was der normale Benutzer kann.

Der Hauptbenutzer gibt beim Log Out den Hauptbenutzer für andere Klienten wieder frei.

Der Hauptbenutzer hat einen Edit und Delete Knopf neben jedem Buch in der Liste. Damit kann ein Buch entweder editiert oder aus der Datenbank gelöscht werden.

In der unteren Reihe gibt es neue Knöpfe für den Hauptbenutzer.

Der grün markierte Knopf zeigt den aktuellen Modus an.

Es können sich Bücher, Kategorien, Autoren und Verlage angesehen werden.

Die Ansicht kann durch das Klicken auf die blauen Knöpfe geändert werden.

Nur im Buch-Modus kann die Auswahl „Kategorie“ als Filter verwendet werden, alle anderen besitzen nur die Texteingabe.

Wird im aktuellen Modus auf den grünen Knopf gedrückt, betritt der Benutzer den jeweiligen Kreieren-Modus.

Es besteht ebenfalls die Möglichkeit die Datenbank in ihren Ursprungszustand zurückzubringen.

Filter, Seitenansicht (Größe, aktuelle Seite) und Order bleiben in den einzelnen Modi für sich bestehen.

Edit Modus:

Mit dem Editieren Knopf kommt der Benutzer in eine neue Ansicht, in der er veränderbare Werte editieren kann.

Editieren von Werten: Durch das Klicken auf den Editieren Knopf kann der Benutzer in einem Dialog den neuen Wert eingeben.

Editieren vom Verlag: Durch das Klicken auf den Editieren Knopf kann der Benutzer in einem Dialog einen neuen Verlag aus einer Auswahl auswählen.

Editieren von Autoren und Kategorien: Durch das Klicken auf den Editieren Knopf kann der Benutzer in einem Dialog die vorhandenen Elemente zwischen den ausgewählten Elementen (linke Liste) und allen nicht ausgewählten Elementen (rechte Liste) wechseln. Bei Bestätigung werden die ausgewählten Elemente in der linken Liste übernommen.

Die Veränderungen können nicht Bestätigt werden, solange ungültige Werte vorhanden sind, ansonsten werden sie in der Datenbank übernommen.

Im Editieren Modus kann sich ebenfalls entschieden werden den Eintrag zu löschen.

Verlassen werden kann der Editieren Modus durch den Exit Button oben links.

Create Modus:

Wenn der Benutzer in den Kreieren-Modus wechselt, erhält er ein leeres Objekt, von dem er alle Werte einstellen kann.

Jeder Wert muss einen gültigen Wert vom Benutzer erhalten.

Wenn ein Buch erstellt wird muss mindestens ein Autor und eine Kategorie verteilt werden.

Die initiale Bewertung kann per Klick auf die Sterne eingestellt werden.

Beim Erstellen eines Autors kann der Alias freigelassen werden, dann wird dieser aus Vor und Nachname erstellt.

Vorgehen / Design-Entscheidungen:

Projektstruktur:

Das Projekt besteht aus den Java Modulen „Client“, „Server“ und „Common“. Hauptsächlich wurde das aus Softwaretechnik bekannte und dort bereits in einer Übung angewandte MVC Modell angewandt.

Dies bedeutet, dass alle Teile des Programms in Model, View und Controller Eingeschalten aufgeteilt wurde.

Datenbank:

Die Datenbank wird beim ersten Start des Servers initialisiert.

Es gibt Bücher mit allen wichtigen Attributen.

Jedes Buch wird von nur einem Verlag veröffentlicht (Publisher\_id wird im jeweiligen Buch als Foreign Key gespeichert)

Jedes Buch kann mehrere Kategorien und Autoren haben (Tabellen book\_authors, book\_categories).

Zusätzlich zu den Büchern gibt es noch Autoren, Verlage und Kategorien, mit ihren jeweiligen Attributen.

Primary Key Buch: ISBN

Primary Key Autor, Publisher, Category: id (auto\_increment für einfachen insert von weiteren Einträgen)

Die Datenbank wird im Programm über einen MySQL Treiber geladen.

Sollte eine Verbindung nicht zustande kommen, so läuft der Server trotzdem weiter, liefert aber keine Daten.

Die Datenbank ist als Singleton implementiert, um konsistenten Zugriff auf die Datenbank zu gewährleisten. Dies wurde deshalb so implementiert, weil es in Softwaretechnik einen Vergleichbaren Übungsaufbau gab, in dem dies so verwendet wurde.

Informationen/ Parameter zu der Datenbank sind im „resources“ Ordner des Server Moduls zu finden und einstellbar.

Common:

Das Modul Common enthält die Schnittstelle für die Kommunikation von Klient und Server.

Unter „resources“ sind in „ServerResources.properties“ alle Informationen über den Server vorhanden. Sie enthält den Namen und Port des Hosts, sowie die einzelnen Ressourcenpfade des Servers. Bei einer Umstrukturierung der Ressourcen können also hier die Pfade geändert werden.

Ebenso kann durch das ändern von „Host.Name“ der Server für unterschiedliche Rechner im Netzwerk erreichbar gemacht werden.

Im „main“ Ordner sind alle Klassen (POJOs) vorhanden, die als Schnittstelle für den Transfer von Daten via JSON notwendig sind.

Der Server nutzt sie, um konkrete Objekte aus den Daten des Servers zu machen und sie in korrektes JSON umzuwandeln.

Der Klient nutzt sie, um eingehende JSON Strings wieder in korrekte Objekte umzuwandeln.

Die Klassen beinhalten alle übertragbaren Informationen: Bücher, Kategorien, Autoren, Verlage.

Jedes Objekt kann mithilfe von JSONGettern und JSONSettern geschrieben und gelesen werden. Die Attribute entsprechen nur denen, die von den Daten in der Datenbank auch verwendet werden sollen.

Es gibt eine Metadaten-Klasse (DBMeta), die dazu verwendet wird, um spezifische Anfragen nach Listen oder Listengrößen zu bewerkstelligen. Sie werden hauptsächlich dafür verwendet, um Objektlisten zu übertragen. Das Attribut „max“ wird dazu verwendet, bei partiellen Ergebnislisten Überblick über die maximalanzahl eines Ergebnissatzes zu erhalten (es werden z.B. alle Bücher von 1 -10 angefragt. Max speichert die maximalanazahl von Büchern, damit festgestellt werden kann, ob das Ende der gesamten Bücherliste schon erreicht ist)

Server:

Im Order „resources“ finden sich alle Parameter für die Verbindung mit der Datenbank, sowie die initialen Datenbankinhalte als SQL Datei.

Der Server wird auf der in Common spezifizierten Server Adresse gestartet.

Der Server verarbeitet alle Anfragen in unterschiedlichen Ressourcen-Klassen (xxxResource).

Diese Klassen können Bearbeitungs-Klassen verwenden, um Anfragen an der Datenbank zu bearbeiten.

Resource Klassen:

UserResource: verarbeitet generelle Anfragen die den Benutzer beim Login betreffen, oder den Server direkt betreffen sollen (reset). Hier wird auch gespeichert, ob es bereits schon einen Hauptbenutzer gibt.

Alle weiteren Ressourcen enthalten die GET, PUT, POST und DELETE Funktionen, die Verwendet werden können, um jeweils Daten von der Datenbank zu holen oder zu verändern.

Es existiert jeweils getAll() und getBySelection(), die eine komplette Liste oder eine partielle Liste als DBMeta zurückgeben.

Die Ressourcen verwenden einen ResourceManager, der die spezifischen Manager enthält.

ResourceManager:

Der ResourceManager enthält generelle Methoden zum Umwandeln von JSON nach Objekten und umgekehrt.

Der ResourceManager muss für alle Ressourcen verfügbar sein. Deshalb wurde sie als Singleton entworfen, sodass alle darin enthaltenen Manager nicht pro Ressource neu erstellt werden müssen.

Ebenso enthält er alle weiteren Manager, die von den einzelnen Ressourcen verwendet werden können, um mit Anfragen an die Datenbank umgehen zu können.

xxxManager:

Jede Resource verwendet ihren eigenen ResourceManager.

Dort werden entweder umgewandelte Objekte in die Datenbank übertragen oder entsprechende Anfragen getätigt, um Daten von der Datenbank zu erhalten.

Jeder Manager besitzt die Funktion, ResultSets des jeweiligen Objekttyps in eine Liste von konkreten Objekten umzuwandeln. Diese kann somit sowohl für Listenanfragen, als auch für Einzelanfragen benutzt werden

Ebenso gibt es die Möglichkeit, Listen der jeweiligen Objekte anzufragen und als DBMeta Objekt zurückzugeben, sowie Anzahlen von Objekten in der Datenbank zu ermitteln.

CategoryManager und AuthorManager erlauben es ebenso, Einträge in die Verbundtabellen book\_authors und book\_categories zu tätigen oder daraus zu löschen.