Datenbanksysteme 2, 5. Übung
Anwendungslogik

Generelles zu dieser Übung

Die Übung baut auf den Ergebnissen von Übung 4 auf. Falls Sie diese noch nicht fertiggestellt haben, sollten Sie dies zuerst erledigen.
Diese Übung ist für 2 Wochen gedacht.

Sie sollten Sich auch in dieser Übung die Arbeit in der Gruppe aufteilen!

Das Ergebnis dieser Übung sowie die relevanten Teile aus Übung 3 (Datenmodell, ActiveRecord-Klassen) werden als erste Bonusaufgabe abgegeben. Bitte geben Sie Ihr Gruppenergebnis als ZIP-Datei (alle relevanten Java-Quellcode-Dateien und SQL-Skripte) per Moodle ab. Eine entsprechende Abgabemöglichkeit ist eingerichtet. Die Abgabe ist bis zum 10.11. möglich. Zur Abgabe müssen Sie zunächst ihre Gruppe in Moodle eintragen.
Um die Bonuspunkte zu erhalten, müssen Sie Ihre Lösung zusätzlich in den Übungen per BBB Ihrem Dozenten vorstellen. Wir vereinbaren dazu nach der Abgabe einen Termin. Zu dem Termin muss die gesamte Gruppe anwesend sein.

Aufgabe 5.1 Erstellung von Anwendungslogik

Auf dem Moodle-Server finden Sie ein Gerüst für ein Java-Programm mit einer einfachen GUI. Kopieren Sie dieses Gerüst und integrieren Sie das Ergebnis von Übung 4 in das Projekt. Dann entwickeln Sie die fehlenden Methoden (mit TODO gekennzeichnet):

GenreManager:
`List<String> getGenres()`. Diese Methode liest alle Genres aus der Datenbank aus und gibt Sie als String-Liste zurück.

MovieManager:
`List<MovieDTO> getMovieList(String search)`. Diese Methode liest alle Filme aus der Datenbank aus, deren Titel **als Teilstring** den übergebenen Parameter enthält (`search`). **Groß-/Kleinschrift** ist dabei zu **ignorieren**.
Die Filme werden dazu als Instanzen einer neuen Klasse zurückgegeben (`MovieDTO`). Die Definition dieser Klasse ist im Projekt enthalten. Wesentlicher Unterschied zur Ihrer Movie-Klasse ist, dass hier auch direkt die Genres und die Charaktere enthalten sind. Für einen Charakter wird dabei auf eine weitere neue Klasse CharacterDTO zurückgegriffen. Gucken Sie zunächst die Definition dieser Klassen an.
Sie müssen in dieser Methode also nicht nur den Movie-Datensatz, sondern auch weitere abhängige Datensätze aus der Datenbank lesen.
Zur Vereinfachung können Sie auf die Methode getMovie zurückgreifen, siehe unten.

`void insertUpdateMovie(MovieDTO movie)`. Diese Methode nimmt ein MovieDTO-Objekt entgegen und fügt es entweder in der Datenbank ein oder aktualisiert den bestehenden Film. Dabei wird die Entscheidung anhand der ID getroffen werden. Wenn diese `null` ist, wird der Film neu eingefügt, ansonsten wird der Film aktualisiert. Auch hier muss die Methode nicht nur den Movie-Datensatz modifizieren, sondern auch die abhängigen (also Genres, Charactere etc.)

`MovieDTO getMovie(long movieId)`. Diese Methode liest einen Filme aus der Datenbank aus. Auch hier müssen zu den Filmen die Genres und Charactere einlesen werden.

Optional können Sie noch die Methode zum Löschen eines Films implementieren.

PersonManager:
`List<String> getPersonList(String text)`. Diese Methode liefert alle in der Datenbank eingetragenen Personennamen, die den Suchstring (`text`) enthalten. Ebenfalls ohne Groß-/Kleinschrift-Berücksichtigung.

Beachten Sie bei der Implementierung folgende Punkte:

- Die Methoden sollen keine SQL-Anweisungen und kein JDBC enthalten. Alle DB-Zugriffe sollen ausschließlich auf die in Übung 4 implementierten Klassen und Factories zurückgreifen. Falls Sie weitere find-Methoden in den Factories benötigen, implementieren Sie diese dort. Einzige Ausnahme sind Anweisungen zur Transaktionskontrolle (Commit und Rollback), die in die Manager-Klassen eingebaut werden sollen. Die ActiveRecord-Klassen sollten keine Transaktionskontrolle enthalten.

- Die DTO-Klassen sind so angelegt, dass eine Person über Ihren Namen identifiziert werden kann. Legen Sie dazu einen passenden Unique-Constraint auf der Personen-Tabelle an, falls dies noch nicht der Fall ist.

- Kapseln Sie Fehler in passenden Exceptions. Der Text zu den Exceptions sollte verständlich sein!

- Achten Sie auf eine passende Transaktionssteuerung.

Testen Sie Ihre Implementierung mit der GUI. Tragen Sie dazu zunächst von Hand (d.h. über SQL Skripte mit dem SQL Developer) einige Genres und Personen in der Datenbank ein, da diese nicht über die Oberfläche ergänzt werden können. Starten Sie dann die GUI und versuchen Sie, einen neuen Film anzulegen, ihn zu suchen und anschließend zu verändern. Tragen Sie Genres und Charaktere zu dem Film ein.

FAQ zu der Aufgabe

1. **Die insertUpdateMovie-Methode kommt mir ziemlich kompliziert vor. Geht das auch einfacher?**

   Sie müssen hier nicht einen genauen Abgleich bzgl. der Genres und Charaktere des Films vornehmen. Man könnte ja z.B. für die Genres genau ermitteln, welche Genres neu hinzugekommen und welche weggefallen sind, und für diese passende Einfüge- und Löschoperationen aufrufen; ebenso für die Charaktere. Dies wird allerdings ziemlich aufwändig, fehleranfällig und schwer zu warten. Daher ist hier eine einfachere Lösung völlig ok: Löschen sie alle Genres und Charaktere zu dem aktuellen Movie und fügen Sie sie passend zu dem DTO-Objekt neu ein.

2. **Was bedeutet das Attribut "Position" in MovieCharacter? Wieso gibt es keine Position im DTO?**

   Die Position bestimmt die Sortierung der Charaktere im Movie, z.B. die Reihenfolge, in der sie im Abspann genannt werden. Da Datensätze in einer Tabelle unsortiert sind und erst durch ORDER BY in eine bestimmte Reihenfolge gebracht werden, wird hier ein Attribut benötigt, nach dem sortiert werden kann. Sie müssen also beim Auslesen

der Charakter-Datensätze "ORDER BY position" verwenden. Im DTO sind die Charaktere durch die Reihenfolge in der Liste sortiert. Hier müssen Sie beim Schreiben der Datensätze einfach eine fortlaufende Nummerierung für "postition" verwenden.

3. Wie genau soll ich mit Exceptions umgehen?

Die ActiveRecord-Klassen sollten gar keine Exceptions abfangen sondern alle Exceptions einfach an die Manager-Klassen durchreichen. Die ActiveRecord-Klassen können aber eigene Fehler als Exception melden. Ein Beispiel dafür wäre die Prüfung des Rückgabewerts von executeUpdate. Wenn dieser nicht dem erwarteten Wert entspricht, werfen sie einfach eine neue Exception mit einem passenden Fehlertext.
Die Manager-Klassen geben Exceptions ebenfalls an die GUI weiter, die GUI enthält bereits einen Dialog, der die Exception anzeigt. Hier müssen Sie nur dafür sorgen, dass die Transaktion im Fehlerfalle abgebrochen wird (rollback). Wenn Sie die in der Vorlesung vorgestellte Vorgehensweise über "finally" verwenden, ist auch hier ein "catch" notwendig.
Auch in den Manager-Klassen können Sie ggf. entdeckte Fehlersituationen über eigene Exceptions melden.