IT4a - AE – Wm – 2016
Florian Petruschke

Inhaltsverzeichnis

1. Warum MongoDB?

Wir haben die Aufgabe bekommen, eine NoSQL-Datenbank zu nutzen, um in Java eine Anwendung zu schreiben, welche Daten aus der Datenbank lesen und in die Datenbank schreiben kann.

Bei mir im Betrieb wird mit Massendaten gearbeitet. Bisher habe ich mich intensiv mit MySQL & SQL auseinandergesetzt. Da wir allerdings für bestimmte Funktionalitäten - mit denen ich bisher noch nicht in Berührung gekommen bin - MongoDB verwenden, war es mir ein Anliegen, mich für die Praxis vorzubereiten. Daher habe ich mich für das Erstellen der Stundenplan-Anwendung für MongoDB als NoSQL-Datenbank entschieden.

2. Voraussetzungen

Voraussetzungen, um das Programm zu verwenden:
(in Klammern die jeweils genutzte Versionsnummer)

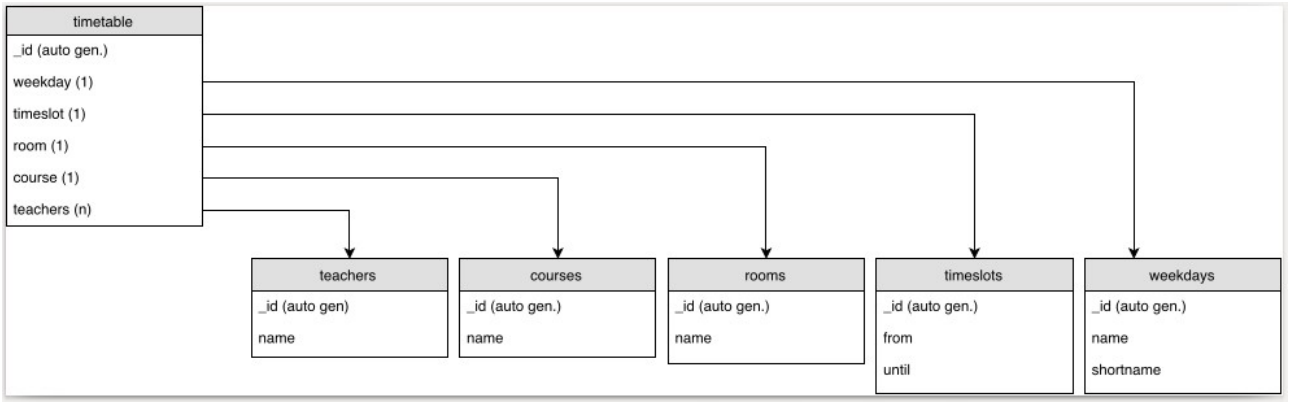
- Java (1.7)
- MongoDB (3.2.9)
- Mongo-Java-Driver (3.4.0-beta1 – im src-Ordner enthalten)

Die Konfiguration für die Verbindung zwischen Anwendung und MongoDB wird in der MongoDbConfig.properties-Datei vorgenommen.

Achtung: Zu Präsentations-Zwecken wurde die Konfigurationsdatei im Repository so geändert, dass eine Verbindung nicht hergestellt werden kann. Bitte ändern Sie die Konfiguration. Alternativ können Sie die .jar ausführen.

3. Datenbank

3.1 Struktur



Die Documents der Collection „timetable“ bestehen jeweils aus Objekten der Collections "weekdays", "timeslots", "teachers", "rooms" und "courses".

3.2 Stammdaten

Ein Stammdaten-Pool (Collection und Documents) wird beim erstmaligen Starten der Anwendung automatisch erzeugt (MongoDbInit). Hierzu muss eventuell die Konfiguration in der MongoDbConfig.properties-Datei angepasst werden.

4. Probleme

Am meisten Probleme hat mir die andere Vorgehensweise von mongoDB bereitet. Es ist schwierig umzudenken, wenn man es gewohnt ist, mit relationalen Datenbanken zu arbeiten und sich überall auf die Relationen zu verlassen. Man muss bei der Arbeit mit mongoDB versuchen, im Kopf zu "de-normalisieren". Außerdem habe ich es als sehr herausfordernd empfunden, die offizielle Dokumentation für den mongodb-java-driver nachzuvollziehen.

5. Arbeitsschritte

- [x] Aufbau der Projektstruktur mit Trennung von Controllern und Gui
- [x] Erstellen des GUIs (Anzeige und Ändern von Daten)
- [x] Installation von mongoDB
- [x] Herstellen einer Verbindung zum lokalen mongoDB server
- [x] Erstellen erster Standard-Abfragen (Select, insert, delete)
- [x] Erstellen einer Tabelle, die mit Daten aus der mongoDB gespeist wird
- [x] Erstellen einer Editierfunktionalität für Tabellenzellen (einzelne Unterrichtseinheiten)
- [x] Hinzufügen von Räumen, Lehrern und Unterrichtsfächern
- [x] Löschen von Räumen, Lehrern und Unterrichtsfächern

6. Blackbox-Testfälle

Voraussetzung für das Testen ist die bestehende mongoDB-Verbindung bei geöffneter Anwendung.

6.1 Anlegen einer Unterrichtseinheit

1. Klicken Sie auf eine leere Tabellenzelle (z.B. Montag, 7:45 - 8:30).
2. In dem sich öffnenden Dialog wählen Sie je einen Kurs, einen Raum und eine Lehrkraft aus.
3. Bestätigen Sie die Eingabe, indem Sie auf "Speichern" klicken
4. In der Tabellenzelle sollte nun der soeben erstelle Datensatz angezeigt werden.

6.2 Editieren einer Unterrichtseinheit

1. Klicke Sie auf eine Tabellenzelle, welche schon gefüllt ist (z.B. Mittwoch, 9:30)
2. In dem sich öffnenden Dialog ändern Sie einen Eintrag (z.B. Raum = 208)
3. Bestätigen Sie die Eingabe, indem Sie auf "Speichern" klicken.
4. In der Tabellenzelle sollte nun der aktualisierte Datensatz angezeigt werden.

6.3 Löschen einer Unterrichtseinheit

1. Klicken Sie auf einen Zelle/Unterrichtseinheit.
2. Klicken Sie in dem sich öffnenden Dialog-Fenster auf den Button "Leeren".
3. Das Dialog-Fenster schließt sich und die eben ausgewählte Zelle wurde geleert.

6.4 Hinzufügen neuer Stammdaten

1. Klicken Sie auf "Stammdaten verwalten".
2. In dem sich öffnenden Dialog geben Sie in das Textfelder unter der Fächer-Liste ein neues Fach ein (z.B. "FE").
3. Bestätigen Sie das neue Fach durch Klicken des darunterliegenden Buttons mit dem "+"-Symbol.
4. Der neu angelegte Fach erscheint als letzter Eintrag in der Fächer-Liste. Das Eingabefeld wurde automatisch geleert.
5. Tragen Sie in das Textfeld unter der Lehrer-Liste Lehrernamen bzw. -kürzel ein (z.B. "Hs").
6. Bestätigen Sie erneut durch Klick auf den darunterliegenden Button mit dem "+"-Symbol. Das Eingabefeld wurde automatisch geleert.
7. Die neue Lehrkraft wird als letzter Eintrag in der Lehrer-Liste angezeigt.
8. Tragen Sie in dem Eingabefeld unter der Raum-Liste einen neuen Raum ein (z.B. "23").
9. Durch Klicken des "+"-Buttons wird der neue Raum in der Raum-Liste angezeigt. Das Eingabefeld wurde automatisch geleert.
10. Klicken Sie auf den Button "Fertig" oder das "X" oben links, um das Dialog-Fenster zu schließen.
11. Legen Sie eine neue Unterrichtseinheit in einer leeren Tabellenzelle an, indem Sie auf die Zelle klicken und die soeben erstellten Daten (z.B. "FE", "Hs", "23") auswählen.
12. Klicke Sie auf "Speichern".
13. Der neue Datensatz erscheint in der von Ihnen ausgewählten Zelle.

6.5 Löschen eines Stammdatensatzes

1. Klicken Sie auf "Stammdaten verwalten".
2. Wählen Sie in dem sich öffnenden Dialog einen Raum aus, indem Sie in der Liste auf den Namen des Raumes klicken.
3. Der Name des Raumes wird in das Eingabefeld unter der Raumliste übernommen.
4. Klicken Sie auf den Button mit dem Mülleimer-Symbol. Der Raum wird nun gelöscht. Das Eingabefeld wurde automatisch geleert.
5. In der Raum-Liste wird der soeben gelöschte Raum nicht mehr angezeigt.
6. Klicken Sie auf den Button "Fertig" oder das "X" oben links, um das Dialog-Fenster zu schließen.

7. Klicken Sie in der Tabelle auf eine schon befüllte Unterrichtseinheit/Zelle.
8. Der soeben gelöschten Raum steht nicht mehr zur Auswahl.

7. Vorbereitung zum Verständnis von mongoDB

Die folgenden Kapitel dienten mir lediglich zur Erarbeitung der Funktionsweise von mongoDB.

7.1 Vokabular

- Collection -> "Tabelle" z.B. db.students
- Document -> "Zeile" z.B. db.students.findOne()
- ObjectId -> beim Erstellen automatisch generierter, eindeutiger Schlüssel (vgl mysql PrimaryKey)

7.2 Basis-Befehle

Befehl	Ergebnis	Typ
use timetable	Wenn noch nicht existent: erstellt die Datenbank "timetable"	CREATE
show collections	zeigt alle Collections	SELECT
db.students.find()	zeigt alle Objekte (Documents) der Collection "Students"	SELECT
db.students.find().pretty()	zeigt alle Objekte (Documents) der Collection "Students" mit Einrückungen	SELECT
db.students.findOne()	zeigt das erste Objekt (Document) der Collection "Students"	SELECT
db.students.insert([{jsonObjectStudent1} ,{jsonObjectStudent2}])	fügt die jeweiligen Objecte zur Collection hinzu	INSERT
db.students.remove({"_id" : ObjectId("idoftheobject123456")})	entfernt das Document mit der ggb. Id aus der Collection "Students"	DELETE
db.students.update({"_id" : ObjectId("idoftheobject123456")},)	aktualisiert den Datensatz	UPDATE
db.students.drop()	löscht die collection "Students"	DROP
db.students.find({"age" : 20}, {"name" : "peter"})	zeigt die Objecte, auf die die Suchmaske/WHERE-Bedingung passt	WHERE
db.students.find($or:{ {"age" : 20}, {"name" : "peter"} })	OR-Bedingung	OR
db.students.find({"age" : {$gt	e/$lt	e/$ne/$e: 20} })
db.students.find({"age" : 20}, {"name" :	Einschränkung der Ergebnismenge (zeigt	WHERE

Befehl	Ergebnis	Typ
1, _id:0 })	nur die Namen, niemals die _id)	
db.students.find({"age" : 20}, {"name" : 1, _id:0 }).limit(3)	Einschränkung: Zeigt maximal 3 Documents	LIMIT
db.students.find({"age" : 20}, {"name" : 1, _id:0 }).skip(2)	Einschränkung: Überspringt die ersten 2 Documents	SKIP
db.students.findOne().explain("executionStats")	zeigt Statistiken zu der Datenbankabfrage	STATUS

7.3 Aggragat-Funktionen

Befehl	Ergebnis	Typ
db.students.aggregate({$group : {_id: "$age", total: {$sum : 1} } })	Summieren einer Ergebnismenge	GROUP + SUM
db.students.aggregate({$group : {_id: "$age", avgAge: {$avg : 1} } })	Durchschnittswert einer Ergebnismenge	GROUP + AVG
db.students.aggregate({$group : {_id: "$age", oldest: {$max : $age} } })	Durchschnittswert einer Ergebnismenge	GROUP + AVG

7.4 Indizes

Indizes sorgen dafür, dass DB-Abfragen effizienter ablaufen. Wann immer z.B. mit einer Bedingung das Suchergebnis gefiltert werden soll, beschleunigt das vorherige Hinzufügen eines Index die Abfrage. Indizes sollten immer nur bei häufig verwendeten Filter-Spalten verwendet werden. Je mehr Indizes bestehen, desto weniger bringt die Verwendung von Indizes.

- db.students.ensureIndex({"age":1}) --> nutze das Alter als Suchindex
- db.students.getIndexes() --> zeigt die angelegten Indizes der Collection