MongoDB und Java

Dependencies

- Maven

```
<dependency>
  <groupId>org.mongodb</groupId>
  <artifactId>mongodb-driver-sync</artifactId>
  <version>4.3.2</version>
</dependency>
```

Verbindungsaufbau mit einem Server

- Mit lokalem MongoDB-Server:

```
MongoClient client = MongoClients.create();
```

- Alternativ:

```
client = MongoClients.create(
                    "mongodb://localhost:27017");
```

Verbindungsaufbau mit Replica-Set

- autom. Ermittlung von Primary Host !

```
client = MongoClients.create(
  "mongodb://host1:27017,host2:27017,...");
```

- Weitere Optionen siehe: https://mongodb.github.io/mongo-java-driver/4.3/driver/tutorials/connect-to-mongodb/

MongoClient

- Nur EINE Instanz von MongoClient nötig
- Limitierungen auf Ressourcen wie Max-Connections gelten pro MongoClient-Instanz
- Mittels MongoClient.close() Ressourcen freigeben!

```
MongoClient client =
                    MongoClients.create(...);
    ...
client.close();
```

MongoDatabase und MongoCollection

- Verbindung zu einer konkreten Datenbank mittels `mongoClient.getDatabase(dbName);`
- Collection für die Datenbank wählen: `MongoCollection<Document> collection = database.getCollection(collName);`

```
MongoDatabase database =
   mongoClient.getDatabase("javatestdb");
MongoCollection<Document> collection =
   database.getCollection("artikel");
```

Document erstellen und einfügen

```
Document artikel =
  new Document("ArtikelNr", 1001)
      .append("Name", "Monitor 22")
      .append("Preis", 159)
      .append("Hersteller", "Acer")
      .append("Spezifikation",
         new Document("Spannung", 240)
             .append("Leistung", 25)
      );

collection.insertOne(artikel);
```

Alternativ: `insertMany(List<Document>);`

Abfragen mittels find()

- `Collection.find()` führt Queries auf die jeweilige Collection durch
- `first()` liefert den ersten Treffer
- `find()`-Methode nimmt als Parameter Filter auf
- `countDocuments()` liefert die Anzahl an Dokumenten in einer Collection

```
Document result =
  (Document)collection.find(
            Filters.eq("ArtikelNr", 1001))
  .first();

System.out.println(
      collection.countDocuments());
```

Iterieren mittels Cursor

- `find().iterator()` liefert eine `FindIterable`-Instanz

```
MongoCursor<Document> cursor =
    collection.find().iterator();

try {
  while (cursor.hasNext()) {
    System.out.println(
            cursor.next().toJson());
  }
} finally {
  cursor.close();
}
```

forEach() auf Ergebnismenge

- `ForEach()` erlaubt das einfache Bearbeiten der Ergebnismenge mittels `mongodb.Block`

```
Consumer<Document> printBlock =
    document -> {
      System.out.println(document.toJson());
    };

Collection
   .find(Filters.gt("Preis", 50))
   .forEach(printBlock);
```

Projektionen und Sortierung

- Wird nicht das gesamte Dokument benötigt, kann dieses mittels Projektionen angepasst werden.
  Dazu wird `projection()` auf `FindIterable` aufgerufen.
  Die Klasse Projections bietet div. Konfigurations-möglichkeiten.
- Sortierung erfolgt mittels `sort()`

```
MongoCursor<Document> cursor =
  collection
    .find()
    .projection(Projections.exclude(
                  "_id", "Spezifikation", "Preis"))
    .sort(Sorts.descending("ArtikelNr"))
    .iterator();
```

Aggregationen

- Werden mittels der Methode `aggregate(List<Bson>)` durchgeführt.
- Die Klassen `Aggregates` und `Accumulators` stellen diverse Hilfsfunktionen bereit

```
collection.aggregate(
  Collections.singletonList(
    Aggregates.group("$Hersteller",
      Accumulators.sum("Gesamtsumme", "$Preis")
    )
  )
).forEach(printBlock);
```

Update von Dokumenten

- `updateOne` und `updateMany` verfügbar
- 1. Parameter: Selektion
  2. Parameter: durchzuführende Änderung
- In der Klasse `Updates` stehen wieder diverse Hilfsmethoden bereit (`set`, `push`, ...)

```
collection.updateOne(
   Filters.eq("ArtikelNr", 1001),
   Updates.set("Preis", 164.90)
);
```

Löschen von Dokumenten

- `deleteOne` und `deleteMany` verfügbar
- 1. Parameter: Selektion
  Returns: `DeleteResult`
- `DeleteResult` liefert mit `getDelectedCount()` die Anzahl der gelöschten Dokumente

```
DeleteResult delResult = collection.deleteOne(
        Filters.eq("ArtikelNr", 2001)
);
System.out.println(
   delResult.getDeletedCount()
   + " Dokument(e) gelöscht!"
);
```