Chapter 3: Admin Backend

Ticket: Migration

Back to the Question

A possible implementation for this lab would be the following:

COPY

```
package mflix;

import com.mongodb.bulk.BulkWriteResult;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.*;
import org.bson.Document;
import org.bson.conversions.Bson;

import java.text.DateFormat;
import java.text.MessageFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

public class Migrator {

  /**
```

```
   * Creates and UpdateOneModel object for each Document that
contains an "imdb.rating" field of
   * non-numerical type into a parsable
   *
   * @param doc - Document object to be updated
   * @return UpdateOneModel operation response object
   */
  private static UpdateOneModel<Document>
transformRating(Document doc) {
    try {
      String imdbRating = doc.get("imdb",
Document.class).getString("rating");

      if (imdbRating == null) {
        return null;
      }

      int rating = 0;
      if (!"".equals(imdbRating)) {
        rating = Integer.valueOf(imdbRating);
      }
      // Update the document based on his _id field
      return new UpdateOneModel<>(
          Filters.eq("_id", doc.getObjectId("_id")),
Updates.set("imdb.rating", rating));
    } catch (NumberFormatException e) {
      System.out.println(
          MessageFormat.format(
              "Could not parse {0} into " + "number: {1}",
doc.get("imdb.rating", e.getMessage())));
    }
    return null;
  }

  /**
   * Creates an UpdateOneModel for each Document object field
`lastupdated` of type string into an
   * update $set to Date type. db.movies.update({_id: doc._id},
{$set: {lastupdated:
   * ISODate(doc.lastupdated)}})
   *
   * @param doc - Document object to get the date transformation
applied to
   * @return UpdateOneModel object or null if no change is
required.
   */
  private static UpdateOneModel<Document> transformDates(Document
doc, DateFormat dateFormat) {

    String lastUpdated = doc.getString("lastupdated");

    try {
      if (lastUpdated != null) {
        return new UpdateOneModel<>(
            Filters.eq("_id", doc.getObjectId("_id")),
            Updates.set("lastupdated",
dateFormat.parse(lastUpdated)));
      }

    } catch (ParseException e) {
      System.out.println(
          MessageFormat.format(
              "String date {0} cannot be parsed using {1} " +
"format: {2}",
              lastUpdated, dateFormat, e.getMessage()));
    }

    return null;
  }

  public static void main(String[] args) {

    System.out.println("Dataset cleanup migration");

    // set your MongoDB Cluster connection string
    String mongoUri = "<YOUR ATLAS CLUSTER URI>";

    // instantiate database and collection objects
    MongoDatabase mflix =
MongoClients.create(mongoUri).getDatabase("mflix");
```

```
    MongoCollection<Document> movies =
mflix.getCollection("movies");
    Bson dateStringFilter =null;
    String datePattern = "";
    // use the same filters as expressed in the `MigrationTest`
unit test
    // to find all documents that need to be updated
    dateStringFilter = Filters.type("lastupdated", "string");
    // define the string pattern to be parsed
    datePattern = "yyyy-MM-dd HH:mm:ss";
    SimpleDateFormat dateFormat = new
SimpleDateFormat(datePattern);

    // create list of bulkWrites to be applied.
    List<WriteModel<Document>> bulkWrites = new ArrayList<>();

    // iterate over the documents and apply the transformations.
    for (Document doc : movies.find(dateStringFilter)) {

      // Apply lastupdate string to date conversion
      WriteModel<Document> updateDate = transformDates(doc,
dateFormat);
      if (updateDate != null) {
        bulkWrites.add(updateDate);
      }
    }

    // same filter has the one found in the unit test for the
rating field.
    Bson ratingStringFilter =
Filters.not(Filters.type("imdb.rating", "number"));

    for (Document doc : movies.find(ratingStringFilter)) {
      // Apply "imdb.rating" string to number conversion
      WriteModel<Document> updateRating = transformRating(doc);
      if (updateRating != null) {
        bulkWrites.add(updateRating);
      }
    }
    // execute the bulk update
    // in this case we don't need the operations to ordered
    BulkWriteOptions bulkWriteOptions = new
BulkWriteOptions().ordered(false);
    if (bulkWrites.isEmpty()) {
      System.out.println("Nothing to update!");
      System.exit(0);
    }

    BulkWriteResult bulkResult = movies.bulkWrite(bulkWrites,
bulkWriteOptions);
    // output the number of updated documents
    System.out.println(
        MessageFormat.format("Updated {0} documents",
bulkResult.getModifiedCount()));
  }
}
```

Proceed to next section