Boutique Travel Boutique

You are now starting a boutique travel travel boutique. Your choice in the hotels you will work with will be very small. To help give your database some structure, you'll be using the npm module mongoose.

Resources

Mongoose Documentation

Mongo Documentation - CRUD

Activity

Set up

Inside this folder

- `touch app.js`
- directory `models` should already be made and have a `seed.js` file
- make sure to export the `seed.js` file in order to be able to import it
- `touch models/hotel.js`
- `npm init`
- `npm install mongoose`

Inside `app.js`

- require mongoose
- configure mongoURI with db called `hotel` : const mongoURI = 'mongodb://localhost:27017/hotel'
- set mongoose connection: `const db = mongoose.connection`
- connect to mongo with `db.on()`, show errors on fail, show disconnection:

```
// ****************************************//
// Everything provided to students in markdown start
// ****************************************//

// Dependencies
const mongoose = require('mongoose')
const db = mongoose.connection

// Config
const mongoURI = 'mongodb://localhost:27017/hotel'

// Models
const Hotel = require('./models/hotel.js')
const hotelSeed = require('./models/seed.js')

// Connect to Mongo
mongoose.connect(mongoURI, { useNewUrlParser: true },
  () => console.log('Mongo running at', mongoURI)
)
```

```
// Error / success
db.on('error', (err) => console.log(err.message + ' is Mongod not running?'))
db.on('connected', () => console.log('mongo connected: ', mongoURI))
db.on('disconnected', () => console.log('mongo disconnected'))

// Hotel.create(hotelSeed, (err, data) => {
//   if (err) console.log(err.message)
//   console.log('added provided hotel data')
// })

// Hotel.collection.drop()

// Hotel.countDocuments({}, (err, data) => {
//   if (err) console.log(err.message)
//   console.log(`There are ${data} hotels in this database`)
// })

// ****************************************//
// Everything provided to students end
// ****************************************//
```

Inside `models/hotel.js`

- require mongoose
- set up the hotel schema
    - name type string, required true, unique true
    - location type string
    - rating type number, max value 5
    - vacancies type boolean
    - tags type array
        - within the array type string
    - rooms

```
  rooms     : [ { roomNumber: Number, size: String, price: Number, booked: Boolean
} ]
```

- set timestamps to true

- use `module.exports` to export this `mongoose.model`

Back to `app.js`

- require `hotel.js` inside the models folder, set to a variable named `Hotel`
- either run with `node app.js` or start nodemon (caution - every time you save the server will restart, possibly causing your commands to go through more times than you'd want)

- Expected output:

```
Mongo running at mongodb://localhost:27017/hotel
Connection made!
```

Seed some Data

- We have some starter data, starter data is often referred to as seed data. Let's load it into your mongo db
- require `models/seed.js` as `hotelSeed` (still working inside your `app.js`)
- Add the following below `db.on('open', ...`

```
Hotel.create( hotelSeed, ( err , data ) => {
      if ( err ) console.log ( err.message )
  console.log( "added provided hotel data" )
  }
);
```

- Once added, comment out or remove the above line or else your db will be populated with duplicates
- **Remember** - when using nodemon, every time you save it'll restart the server and run everything you have
- If you accidentally made duplicates, you can drop your whole database and start again
    - add and run the following line once, comment out or remove when done

```
Hotel.collection.drop();
```

Check for the right number of hotels:

```
Hotel.count({} , (err , data)=> {
   if ( err ) console.log( err.message );
    console.log ( `There are ${data} hotels in this database` );
});
```

There should be 12 hotels in the database.

Using Mongoose to CRUD our data

With each prompt, complete it, then comment it out - write all your code inside `app.js`
Mongo Documentation

C

- Create a hotel using the above schema
    - add your mongoose command inside app.js. Once it works, comment it out.

    - There should now be 13 hotels in the database.

R

- with each prompt, complete it, then comment it out
    - let's find all our hotels and `console.log` them
    - find all the hotels but only return the hotel name and `console.log` them
    - find just your hotel using a search parameter that would only match your hotel
    - find all the hotels that have vacancies, also exclude ratings.

D

- turns out Hotelicopter was an April Fool's prank! Let's delete that one from our database
- Hilbert's Hotel is getting terrible ratings all it does is give everyone headaches and no room service. Let's just remove that one as well
- The hotel in the `Colorado Rockies` has been closed for undisclosed reasons. Delete this hotel too

U

Note be sure to console log the updated document (hint

- The Great Northern's rating is only a 3! Update that to be a 5
- Motel Bambi is now fully booked, change the vacancies to false
- Things are on the decline for the Motel in `'White Bay, Oregon'`, change the rating to 2

Hungry for More (choose any)

- install express
- create an app.get route that `res.send` all the hotels to the browsers to be viewed as json
- create a route '/:id' that takes the hotel id as a parameter and then displays just the hotel with the matching id

- Don't need express for the following:
    - update the prices of each of the rooms at Fawlty Towers

    - find the hotel with an `indoor pool` as a tag
    - The PR firm for the Hyperion Hotel has demanded that 'crime' is taken off as a keyword, remove that keyword

Wildly Ravenous For Even More

- Use EJS to make views of your data