## Connect to MongoDB

Open /app.js (in the root of your project) and copy the following text below where you declare the Express application object (after the line var app = express();). Replace the database url string ('insert\_your\_database\_url\_here') with the location URL representing your own database (i.e. using your MongoDB URI).

//Set up mongoose connection

var mongoose = require('mongoose');

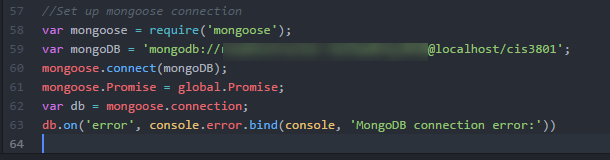
var mongoDB = 'insert\_your\_database\_url\_here';

mongoose.connect(mongoDB);

mongoose.Promise = global.Promise;

var db = mongoose.connection;

db.on('error', console.error.bind(console, 'MongoDB connection error:'));



## Defining the Rasmussen Library Schema

Using Atom, create a separate module for each model, as discussed above. Create a folder called "models" in the project root (/models) - this should be in the sample folder and then create separate files for each of the models:

/sample //the project root

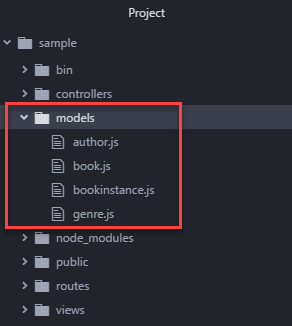
/models

author.js

book.js

bookinstance.js

genre.js



Extract files from models.zip to this folder.

## Populate sample database

We'll now run an independent script to create items of each type:

Download the file populatedb.js inside your sample directory (in the same level as package.json).

Enter the following commands in the project root to install the async module that is required by the script.

sudo npm install async --save

Run the script using node in your command prompt, passing in the URL of your MongoDB database (the same one you replaced the insert\_your\_database\_url\_here placeholder with, inside app.js earlier):

sudo node populatedb <your mongodb url>​​​​

Your mongodb url would be: mongodb://[username]:[password]@localhost/cis3801

The script should run through to completion, displaying items as it creates them in the terminal.

## Creating Routes or URLs

The URLs that we’ll need for our pages are listed below, where object is replaced by the name of each of our models (book, bookinstance, genre, author), objects is the plural of object, and id is the unique instance field (\_id) that is given to each Mongoose model instance by default.

* catalog/ — The home/index page.
* catalog/<objects>/ — The list of all books, bookinstances, genres, or authors (e.g. /catalog/books/, /catalog/genres/, etc.)
* catalog/<object>/<id> — The detail page for a specific book, bookinstance, genre, or author with the given \_id field value (e.g. /catalog/book/584493c1f4887f06c0e67d37).
* catalog/<object>/create — The form to create a new book, bookinstance, genre, or author (e.g. /catalog/book/create).
* catalog/<object>/<id>/update — The form to update a specific book, bookinstance, genre, or author with the given \_id field value (e.g. /catalog/book/584493c1f4887f06c0e67d37/update).
* catalog/<object>/<id>/delete — The form to delete a specific book, bookinstance, genre, author with the given \_id field value (e.g. /catalog/book/584493c1f4887f06c0e67d37/delete).

## Create the route-handler callback functions

Before routes are defined, we'll first create the callback functions that they will invoke. The callbacks will be stored in separate "controller" modules for Books, BookInstances, Genres, and Authors.

Start by creating a folder for our controllers in the project root (/controllers) and then unzip and extract the controllers.zip file to this folder.

/sample //the project root

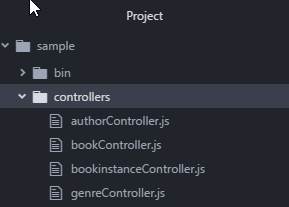
/controllers

authorController.js

bookController.js

bookinstanceController.js

genreController.js



## Create the catalog route module

Now, you’ll create routes for all the URLs needed by the sample website, which will call the controller functions we defined in the previous section.

Unzip and extract routes.zip to this folder.

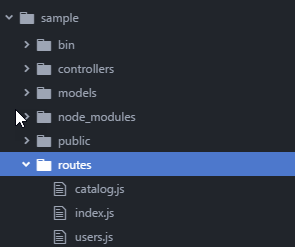
/sample //the project root

/routes

index.js

users.js

catalog.js



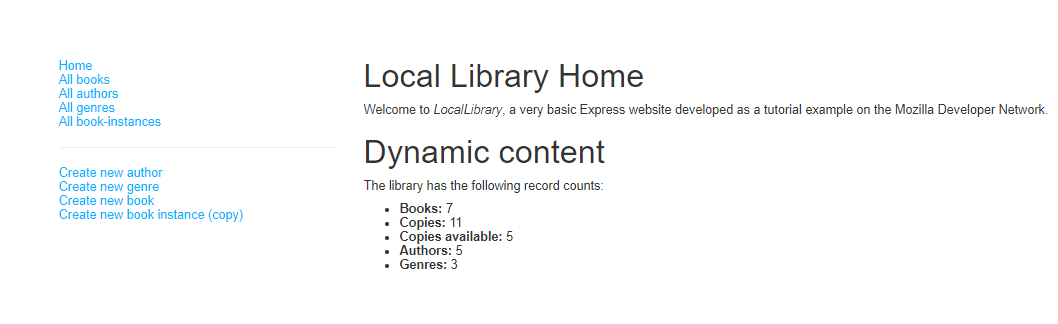
Test the app by typing:

./bin/www

You should be able to navigate to the following URLs. Verify that you don't get an error page (HTTP 404).

* <http://localhost:3000/>
* <http://localhost:3000/catalog>
* <http://localhost:3000/catalog/books>
* <http://localhost:3000/catalog/bookinstances/>
* <http://localhost:3000/catalog/authors/>
* <http://localhost:3000/catalog/genres/>
* <http://localhost:3000/catalog/book/5846437593935e2f8c2aa226>
* <http://localhost:3000/catalog/book/create>

If you see a root page like the one below, you’ve successfully installed the application.



*The full copy of the sample code is available in sample.zip. However, note that simply copying the entire folder will not create the new application. You must follow the steps outlined in this document.*

Reference: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs>