

Developing Express example towards a restful server.js

restful server

In the previous sections have shown the process of setting up general node projects and specifically express-generated projects and getting the platform working.

Now look at how the express application is developed to produce a restful interface.

The server should respond to the url with which it is addressed and the http command GET, POST, DELETE. This is done by setting up routes.

routes

To preserve the previous working (empty shell) as an easy starting point for further projects. Duplicate folder express_app1 to express_app2. Change the directory in the terminal to this folder.

Start the server running in a container named myapp2

```
docker run --name myapp2 -p 3000:3000 --mount
type=bind,source=$(pwd)/myapp,target=/usr/src/app -it dt/express-
development
```

app.js

Take a look at app.js section by section.

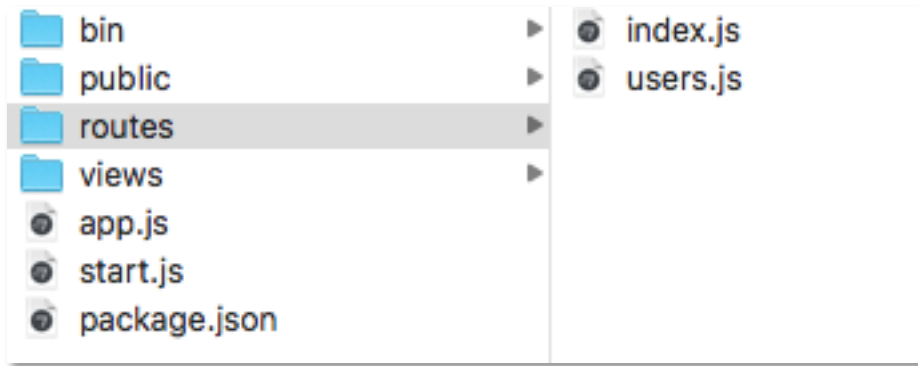
Node modules essential to operation are imported using the require function. These have already been called into the environment by npm install acting on the list of dependencies in package.json; so, no downloading is happening here.

```
http-errors
express
path
cookie-parser
morgan
```

```
var createError = require("http-errors");
var express = require("express");
var path = require("path");
var cookieParser = require("cookie-parser");
var logger = require("morgan");
```

Import the modules which are in the routes directory as index.js and users.js. These routes will be edited and used later.

```
var indexRouter = require("./routes/index");
var usersRouter = require("./routes/users");
```



routes

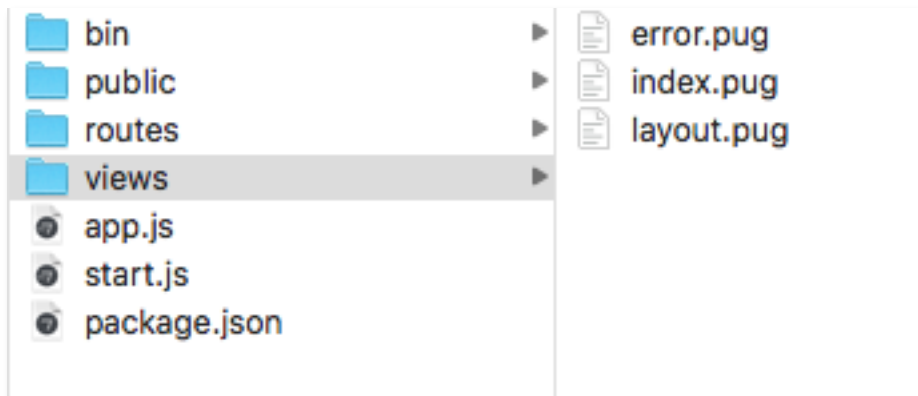
Create an app object using the express constructor. Then set up view templates. First add the views directory to the app path. Then set the view engine to the choice made at the time of generating the project, which was [pug](#).

```
var app = express();
```

```
// view engine setup
```

```
app.set("views", path.join(__dirname, "views"));
```

```
app.set("view engine", "pug");
```



routes

Next functions call `app.use` to add middleware which has already been imported into variables. So for example `logger` is an object imported from `morgan`.

The use of [express static](#) allows the server to serve static web pages from the folder listed. That allows for html pages, images, audio files etc to be served from the public directory. Other directories could also be added as required.

```
app.use(logger("dev"));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, "public")));
```

Week 8

Route handling code which has already been imported to variables `indexRouter` and `usersRouter` is added to the request handling chain of the app.

Requests with a prefix `/users` such as `users/list` will be handled by the routes set up in the `users` file. Other files can be added to set up a complex rest API

```
app.use("/", indexRouter);
app.use("/users", usersRouter);
```

Errors from asynchronous functions invoked by route handlers must be passed to the next function for [error handling](#).

```
// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get("env") === "development" ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render("error");
});
```

Finally the app is fully configured. Exporting it allows it to be imported by `.bin/www` and the application can be started.

```
module.exports = app;
```

Routes

The contents of `index.js` have already been imported to `app`. When a GET request `/` is recieved a simple home page is rendered.

```
var express = require("express");
var router = express.Router();

/* GET home page. */
router.get("/", function(req, res, next) {
  res.render("index", { title: "Express with Nodemon" });
});

module.exports = router;
```

The home page is rendered by `response.render` written as `res.render(view [, locals] [, callback])` The view is defined by the template in `views/index.pug`

Week 8

extends layout

block content

```
h1= title
p Welcome to #{title}
```

In a similar way any HTTP requests which come in to '/users' such as '/users/fred' or '/users/jane' will be served by the routes described in users.js

users.js does not render a page into the response, but simply sends a text message.

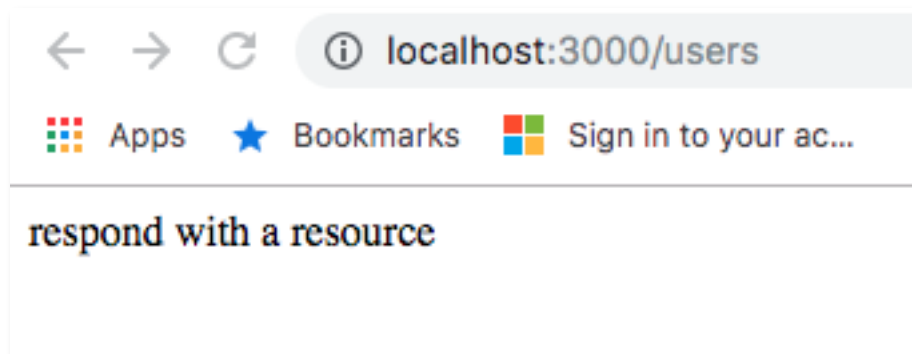
```
var express = require("express");
var router = express.Router();

/* GET users listing. */
router.get("/", function(req, res, next) {
  res.send("respond with a resource");
});
```

```
module.exports = router;
```

To see the output of the users route, with myapp2 running, browse to:

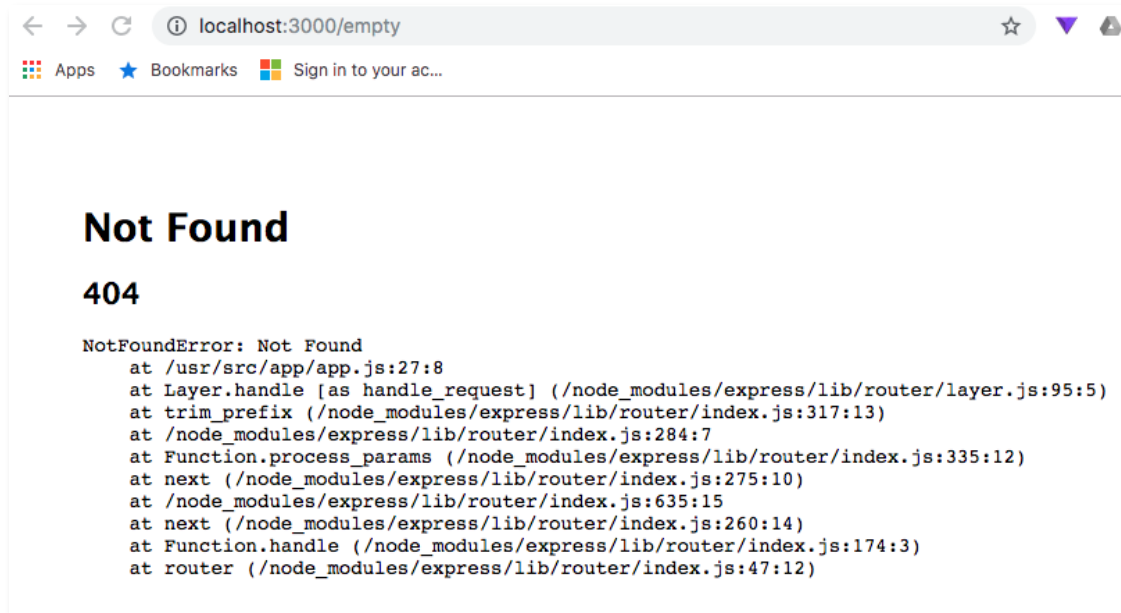
localhost:3000/users



user

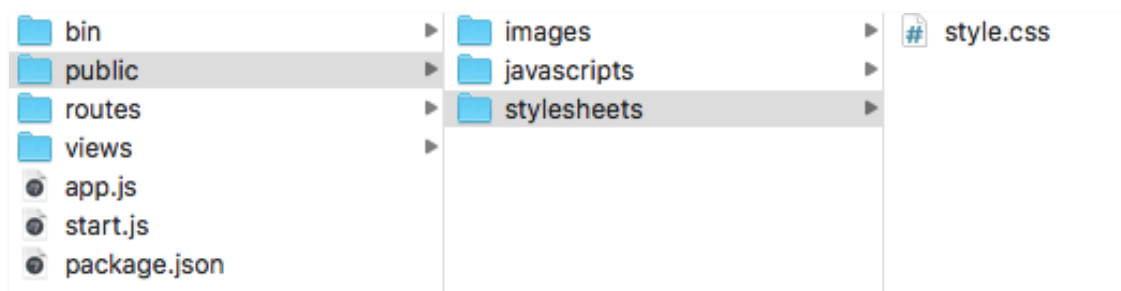
To see a 404 error browse to:

localhost:3000/empty



empty

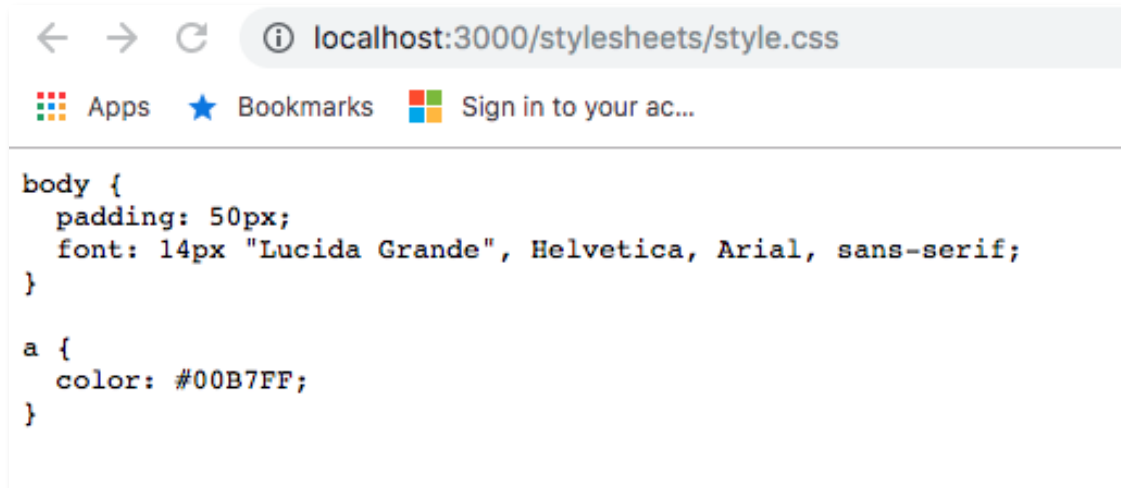
To see a static file delivered from the public folder



public

browse to:

<http://localhost:3000/stylesheets/style.css>

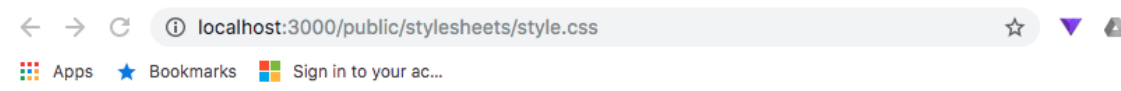


browse inside public

This shows the stylesheet in the public folder. Note that the word public is not part of the url.

`http://localhost:3000/public/stylesheets/style.css`

Yeilds a 404 error!



Not Found

404

```
NotFoundError: Not Found
    at /usr/src/app/app.js:27:8
    at Layer.handle [as handle_request] (/node_modules/express/lib/router/layer.js:95:5)
    at trim_prefix (/node_modules/express/lib/router/index.js:317:13)
    at /node_modules/express/lib/router/index.js:284:7
    at Function.process_params (/node_modules/express/lib/router/index.js:335:12)
    at next (/node_modules/express/lib/router/index.js:275:10)
    at /node_modules/express/lib/router/index.js:635:15
    at next (/node_modules/express/lib/router/index.js:260:14)
    at Function.handle (/node_modules/express/lib/router/index.js:174:3)
    at router (/node_modules/express/lib/router/index.js:47:12)
```

browse using public in url

Database and controller

Mongo database described on [docker hub](#) as a free and open-source cross-platform document-oriented database program. Classified as a **NoSQL** database program, MongoDB uses **JSON**-like documents with **schemata**.

Week 8

To get an official docker image of MongoDB:

```
docker pull mongo
```

Terminal output:

```
Using default tag: latest
latest: Pulling from library/mongo
f7277927d38a: Pull complete
8d3eac894db4: Pull complete
edf72af6d627: Pull complete
3e4f86211d23: Pull complete
5747135f14d2: Pull complete
f56f2c3793f6: Pull complete
f8b941527f3a: Pull complete
4000e5ef59f4: Pull complete
ad518e2379cf: Pull complete
8f879b25c44a: Pull complete
cff21415a903: Pull complete
745d8bb53b0b: Pull complete
0b2aa35fb931: Pull complete
Digest:
sha256:33ec77dc606c4ae24e6cfcac832ff71e7fb6c6943d5b62b9a7ae94763de088bb
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest
```

Mongo-express is described on [docker hub](#) as web-based MongoDB admin interface written in Node.js, Express.js, and Bootstrap3.

To get an official docker image of mongo-express:

```
docker pull mongo-express
```

Terminal output:

```
Using default tag: latest
latest: Pulling from library/mongo-express
e7c96db7181b: Already exists
5b5ce4d3d5f2: Pull complete
ac289eff61bc: Pull complete
28fbd2f3252d: Pull complete
355aefc57dbb: Pull complete
be052a1dc11e: Pull complete
0150ab596ec1: Pull complete
366642a63bb1: Pull complete
Digest:
sha256:c62cd3ce334dcabcea72ff11c993f51d3718a0dbf0037d0f71282102db0d59c8
Status: Downloaded newer image for mongo-express:latest
docker.io/library/mongo-express:latest
```

Docker Run and compose

It would be possible to run mongo with a line such as:

```
docker run --name some-mongo -d mongo:tag
```

and mongo express as

```
docker run --network some-network -e ME_CONFIG_MONGODB_SERVER=some-mongo -p 8081:8081 mongo-express
```

This introduces the [docker network](#) for connecting between separate containers. This is configured from [docker run](#) with `--network`.

However when two or more containers are to be run together it is preferable to use the [docker compose tool](#) rather than run.

[Docker compose](#) can be used with these (and other) command line commands

docker-compose	command	Purpose
docker-compose	build	Services are built once and then tagged
docker-compose	up	docker-compose
docker-compose	pause	Pauses running containers of a service.
docker-compose	unpause	Unpauses paused containers of a service.
docker-compose	stop	Stops running containers without removing them.
docker-compose	start	Starts existing containers for a service.
docker-compose	down	Stops containers and removes containers, networks, volumes, and images created by up.
docker-compose	restart	Restarts all stopped and running services.
docker-compose	kill	Forces running containers to stop by sending a SIGKILL signal.
docker-compose	ps	Lists containers.
docker-compose	rm	Removes stopped service containers.

The build stage needs compose .yaml file and a .dockerfile. Because we have pulled images directly from the hub with no further customisation there is no need for a build stage here.

Week 8

The up stage uses a .yaml file to describe the configurations which would have been in the docker run line.

Save stack.yml in a new folder mongo1 and open a terminal window directed to this folder.

```
# Use root/example as user/password credentials
version: "3.1"
```

```
services:
  mongo:
    image: mongo
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: example

  mongo-express:
    image: mongo-express
    restart: always
    ports:
      - 8081:8081
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: root
      ME_CONFIG_MONGODB_ADMINPASSWORD: example
```

This will start two services mongo and mongo-express.

```
docker-compose -f stack.yml up
```

The -f option is used to specify an alternate compose file to the default which would be named as docker-compose.yml

```
env_file: .env
```

There is lots of terminal output.

```
Creating network "mongo1_default" with the default driver
Creating mongo1_mongo_1 ... done
Creating mongo1_mongo-express_1 ... done
Attaching to mongo1_mongo_1, mongo1_mongo-express_1
mongo-express_1 | Waiting for mongo:27017...
mongo-express_1 | /docker-entrypoint.sh: connect: Connection refused
mongo-express_1 | /docker-entrypoint.sh: line 14: /dev/tcp/mongo/27017:
Connection refused
mongo_1          | 2019-08-12T22:54:03.614+0000 I STORAGE [main] Max cache
overflow file size custom option: 0
mongo_1          | about to fork child process, waiting until server is ready
for connections.
mongo_1          | forked process: 25
mongo_1          | 2019-08-12T22:54:03.623+0000 I CONTROL [main] *****
SERVER RESTARTED *****
```

Week 8

```
mongo_1      | 2019-08-12T22:54:03.640+0000 I CONTROL [main]
Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --
sslDisabledProtocols 'none'
mongo_1      | 2019-08-12T22:54:03.655+0000 I CONTROL [initandlisten]
MongoDB starting : pid=25 port=27017 dbpath=/data/db 64-bit host=f04785dee1c8
mongo_1      | 2019-08-12T22:54:03.655+0000 I CONTROL [initandlisten] db
version v4.0.12
mongo_1      | 2019-08-12T22:54:03.655+0000 I CONTROL [initandlisten]
git version: 5776e3cbf9e7afe86e6b29e22520ffb6766e95d4
mongo_1      | 2019-08-12T22:54:03.655+0000 I CONTROL [initandlisten]
OpenSSL version: OpenSSL 1.0.2g 1 Mar 2016
mongo_1      | 2019-08-12T22:54:03.655+0000 I CONTROL [initandlisten]
allocator: tcmalloc
mongo_1      | 2019-08-12T22:54:03.655+0000 I CONTROL [initandlisten]
modules: none
mongo_1      | 2019-08-12T22:54:03.655+0000 I CONTROL [initandlisten]
build environment:
mongo_1      | 2019-08-12T22:54:03.655+0000 I CONTROL [initandlisten]
distmod: ubuntu1604
mongo_1      | 2019-08-12T22:54:03.655+0000 I CONTROL [initandlisten]
distarch: x86_64
mongo_1      | 2019-08-12T22:54:03.655+0000 I CONTROL [initandlisten]
target_arch: x86_64
mongo_1      | 2019-08-12T22:54:03.655+0000 I CONTROL [initandlisten]
options: { net: { bindIp: "127.0.0.1", port: 27017, ssl: { mode: "disabled" }
}, processManagement: { fork: true, pidFilePath: "/tmp/docker-entrypoint-
temp-mongod.pid" }, systemLog: { destination: "file", logAppend: true, path:
"/proc/1/fd/1" } }
mongo_1      | 2019-08-12T22:54:03.656+0000 I STORAGE [initandlisten]
mongo_1      | 2019-08-12T22:54:03.657+0000 I STORAGE [initandlisten] **
WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger
storage engine
mongo_1      | 2019-08-12T22:54:03.657+0000 I STORAGE [initandlisten] **
See http://dochub.mongodb.org/core/prodnotes-filesystem
mongo_1      | 2019-08-12T22:54:03.657+0000 I STORAGE [initandlisten]
wiredtiger_open config:
create,cache_size=487M,cache_overflow=(file_max=0M),session_max=20000,evictio
n=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enab
led=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idl
e_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
mongo_1      | 2019-08-12T22:54:04.366+0000 I STORAGE [initandlisten]
WiredTiger message [1565650444:366647][25:0x7f1d6f184a80], txn-recover: Set
global recovery timestamp: 0
mongo_1      | 2019-08-12T22:54:04.379+0000 I RECOVERY [initandlisten]
WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
mongo_1      | 2019-08-12T22:54:04.398+0000 I CONTROL [initandlisten]
mongo_1      | 2019-08-12T22:54:04.398+0000 I CONTROL [initandlisten] **
WARNING: Access control is not enabled for the database.
mongo_1      | 2019-08-12T22:54:04.398+0000 I CONTROL [initandlisten] **
Read and write access to data and configuration is unrestricted.
```

Week 8

```
mongo_1          | 2019-08-12T22:54:04.398+0000 I CONTROL [initandlisten]
mongo_1          | 2019-08-12T22:54:04.400+0000 I STORAGE [initandlisten]
createCollection: admin.system.version with provided UUID: e0b1a830-686e-
4349-9840-24905dfefda1
mongo_1          | 2019-08-12T22:54:04.416+0000 I COMMAND [initandlisten]
setting featureCompatibilityVersion to 4.0
mongo_1          | 2019-08-12T22:54:04.421+0000 I STORAGE [initandlisten]
createCollection: local.startup_log with generated UUID: 6b2db14e-dd13-4f4c-
95ea-10f9b1a35f41
mongo_1          | 2019-08-12T22:54:04.437+0000 I FTDC [initandlisten]
Initializing full-time diagnostic data capture with directory
'/data/db/diagnostic.data'
mongo_1          | 2019-08-12T22:54:04.440+0000 I NETWORK [initandlisten]
waiting for connections on port 27017
mongo_1          | 2019-08-12T22:54:04.440+0000 I STORAGE
[LogicalSessionCacheRefresh] createCollection: config.system.sessions with
generated UUID: b77766bf-9e18-462e-9230-4fac1e992ffa
mongo_1          | child process started successfully, parent exiting
mongo_1          | 2019-08-12T22:54:04.469+0000 I INDEX
[LogicalSessionCacheRefresh] build index on: config.system.sessions
properties: { v: 2, key: { lastUse: 1 }, name: "lsidTTLIndex", ns:
"config.system.sessions", expireAfterSeconds: 1800 }
mongo_1          | 2019-08-12T22:54:04.469+0000 I INDEX
[LogicalSessionCacheRefresh] building index using bulk method; build may
temporarily use up to 500 megabytes of RAM
mongo_1          | 2019-08-12T22:54:04.471+0000 I INDEX
[LogicalSessionCacheRefresh] build index done. scanned 0 total records. 0
secs
mongo_1          | 2019-08-12T22:54:04.539+0000 I NETWORK [listener]
connection accepted from 127.0.0.1:59870 #1 (1 connection now open)
mongo_1          | 2019-08-12T22:54:04.541+0000 I NETWORK [conn1] received
client metadata from 127.0.0.1:59870 conn1: { application: { name: "MongoDB
Shell" }, driver: { name: "MongoDB Internal Client", version: "4.0.12" }, os:
{ type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "16.04" } }
mongo_1          | 2019-08-12T22:54:04.547+0000 I NETWORK [conn1] end
connection 127.0.0.1:59870 (0 connections now open)
mongo-express_1 | Mon Aug 12 22:54:04 UTC 2019 retrying to connect to
mongo:27017 (2/5)
mongo-express_1 | /docker-entrypoint.sh: connect: Connection refused
mongo-express_1 | /docker-entrypoint.sh: line 14: /dev/tcp/mongo/27017:
Connection refused
mongo_1          | 2019-08-12T22:54:04.639+0000 I NETWORK [listener]
connection accepted from 127.0.0.1:59874 #2 (1 connection now open)
mongo_1          | 2019-08-12T22:54:04.640+0000 I NETWORK [conn2] received
client metadata from 127.0.0.1:59874 conn2: { application: { name: "MongoDB
Shell" }, driver: { name: "MongoDB Internal Client", version: "4.0.12" }, os:
{ type: "Linux", name: "Ubuntu", architecture: "x86_64", version: "16.04" } }
mongo_1          | 2019-08-12T22:54:04.686+0000 I STORAGE [conn2]
createCollection: admin.system.users with generated UUID: 3aafae4a-e9d1-46da-
883e-ad25a38b54fb
```

Week 8

```
mongo_1      | Successfully added user: {
mongo_1      |   "user" : "root",
mongo_1      |   "roles" : [
mongo_1      |     {
mongo_1      |       "role" : "root",
mongo_1      |       "db" : "admin"
mongo_1      |     }
mongo_1      |   ]
mongo_1      | }
mongo_1      | 2019-08-12T22:54:04.707+0000 E - [main] Error
saving history file: FileOpenFailed: Unable to open() file
/home/mongodb/.dbshell: No such file or directory
mongo_1      | 2019-08-12T22:54:04.709+0000 I NETWORK [conn2] end
connection 127.0.0.1:59874 (0 connections now open)
mongo_1      |
mongo_1      | /usr/local/bin/docker-entrypoint.sh: ignoring /docker-
entrypoint-initdb.d/*
mongo_1      |
mongo_1      | 2019-08-12T22:54:04.733+0000 I STORAGE [main] Max cache
overflow file size custom option: 0
mongo_1      | 2019-08-12T22:54:04.734+0000 I CONTROL [main] *****
SERVER RESTARTED *****
mongo_1      | 2019-08-12T22:54:04.737+0000 I CONTROL [main]
Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --
sslDisabledProtocols 'none'
mongo_1      | killing process with pid: 25
mongo_1      | 2019-08-12T22:54:04.741+0000 I CONTROL
[signalProcessingThread] got signal 15 (Terminated), will terminate after
current cmd ends
mongo_1      | 2019-08-12T22:54:04.741+0000 I NETWORK
[signalProcessingThread] shutdown: going to close listening sockets...
mongo_1      | 2019-08-12T22:54:04.741+0000 I NETWORK
[signalProcessingThread] removing socket file: /tmp/mongodb-27017.sock
mongo_1      | 2019-08-12T22:54:04.742+0000 I CONTROL
[signalProcessingThread] Shutting down free monitoring
mongo_1      | 2019-08-12T22:54:04.742+0000 I FTDC
[signalProcessingThread] Shutting down full-time diagnostic data capture
mongo_1      | 2019-08-12T22:54:04.742+0000 I STORAGE
[signalProcessingThread] WiredTigerKVEngine shutting down
mongo_1      | 2019-08-12T22:54:04.746+0000 I STORAGE
[signalProcessingThread] Shutting down session sweeper thread
mongo_1      | 2019-08-12T22:54:04.746+0000 I STORAGE
[signalProcessingThread] Finished shutting down session sweeper thread
mongo_1      | 2019-08-12T22:54:04.834+0000 I STORAGE
[signalProcessingThread] shutdown: removing fs lock...
mongo_1      | 2019-08-12T22:54:04.835+0000 I CONTROL
[signalProcessingThread] now exiting
mongo_1      | 2019-08-12T22:54:04.835+0000 I CONTROL
[signalProcessingThread] shutting down with code:0
mongo-express_1 | Mon Aug 12 22:54:05 UTC 2019 retrying to connect to
```

```

mongo:27017 (3/5)
mongo-express_1 | /docker-entrypoint.sh: connect: Connection refused
mongo-express_1 | /docker-entrypoint.sh: line 14: /dev/tcp/mongo/27017:
Connection refused
mongo_1 |
mongo_1 | MongoDB init process complete; ready for start up.
mongo_1 |
mongo_1 | 2019-08-12T22:54:05.787+0000 I STORAGE [main] Max cache
overflow file size custom option: 0
mongo_1 | 2019-08-12T22:54:05.790+0000 I CONTROL [main]
Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --
sslDisabledProtocols 'none'
mongo_1 | 2019-08-12T22:54:05.794+0000 I CONTROL [initandlisten]
MongoDB starting : pid=1 port=27017 dbpath=/data/db 64-bit host=f04785dee1c8
mongo_1 | 2019-08-12T22:54:05.794+0000 I CONTROL [initandlisten] db
version v4.0.12
mongo_1 | 2019-08-12T22:54:05.794+0000 I CONTROL [initandlisten]
git version: 5776e3cbf9e7afe86e6b29e22520ffb6766e95d4
mongo_1 | 2019-08-12T22:54:05.794+0000 I CONTROL [initandlisten]
OpenSSL version: OpenSSL 1.0.2g 1 Mar 2016
mongo_1 | 2019-08-12T22:54:05.794+0000 I CONTROL [initandlisten]
allocator: tcmalloc
mongo_1 | 2019-08-12T22:54:05.794+0000 I CONTROL [initandlisten]
modules: none
mongo_1 | 2019-08-12T22:54:05.794+0000 I CONTROL [initandlisten]
build environment:
mongo_1 | 2019-08-12T22:54:05.794+0000 I CONTROL [initandlisten]
distmod: ubuntu1604
mongo_1 | 2019-08-12T22:54:05.794+0000 I CONTROL [initandlisten]
distarch: x86_64
mongo_1 | 2019-08-12T22:54:05.794+0000 I CONTROL [initandlisten]
target_arch: x86_64
mongo_1 | 2019-08-12T22:54:05.794+0000 I CONTROL [initandlisten]
options: { net: { bindIpAll: true }, security: { authorization: "enabled" } }
mongo_1 | 2019-08-12T22:54:05.795+0000 I STORAGE [initandlisten]
Detected data files in /data/db created by the 'wiredTiger' storage engine,
so setting the active storage engine to 'wiredTiger'.
mongo_1 | 2019-08-12T22:54:05.795+0000 I STORAGE [initandlisten]
mongo_1 | 2019-08-12T22:54:05.795+0000 I STORAGE [initandlisten] **
WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger
storage engine
mongo_1 | 2019-08-12T22:54:05.796+0000 I STORAGE [initandlisten] **
See http://dochub.mongodb.org/core/prodnotes-filesystem
mongo_1 | 2019-08-12T22:54:05.796+0000 I STORAGE [initandlisten]
wiredtiger_open config:
create,cache_size=487M,cache_overflow=(file_max=0M),session_max=20000,evictio
n=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enab
led=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idl
e_time=100000),statistics_log=(wait=0),verbose=(recovery_progress),
mongo-express_1 | Mon Aug 12 22:54:06 UTC 2019 retrying to connect to

```

Week 8

```
mongo:27017 (4/5)
mongo-express_1 | /docker-entrypoint.sh: connect: Connection refused
mongo-express_1 | /docker-entrypoint.sh: line 14: /dev/tcp/mongo/27017:
Connection refused
mongo_1          | 2019-08-12T22:54:07.027+0000 I STORAGE [initandlisten]
WiredTiger message [1565650447:27192][1:0x7f0b41e9fa80], txn-recover: Main
recovery loop: starting at 1/27264 to 2/256
mongo_1          | 2019-08-12T22:54:07.128+0000 I STORAGE [initandlisten]
WiredTiger message [1565650447:128795][1:0x7f0b41e9fa80], txn-recover:
Recovering log 1 through 2
mongo_1          | 2019-08-12T22:54:07.204+0000 I STORAGE [initandlisten]
WiredTiger message [1565650447:204010][1:0x7f0b41e9fa80], txn-recover:
Recovering log 2 through 2
mongo_1          | 2019-08-12T22:54:07.255+0000 I STORAGE [initandlisten]
WiredTiger message [1565650447:255927][1:0x7f0b41e9fa80], txn-recover: Set
global recovery timestamp: 0
mongo_1          | 2019-08-12T22:54:07.273+0000 I RECOVERY [initandlisten]
WiredTiger recoveryTimestamp: Ts: Timestamp(0, 0)
mongo_1          | 2019-08-12T22:54:07.306+0000 I FTDC [initandlisten]
Initializing full-time diagnostic data capture with directory
'/data/db/diagnostic.data'
mongo_1          | 2019-08-12T22:54:07.308+0000 I NETWORK [initandlisten]
waiting for connections on port 27017
mongo-express_1 | Mon Aug 12 22:54:07 UTC 2019 retrying to connect to
mongo:27017 (5/5)
mongo_1          | 2019-08-12T22:54:07.626+0000 I NETWORK [listener]
connection accepted from 172.19.0.3:40940 #1 (1 connection now open)
mongo_1          | 2019-08-12T22:54:07.627+0000 I NETWORK [conn1] end
connection 172.19.0.3:40940 (0 connections now open)
mongo-express_1 | Welcome to mongo-express
mongo-express_1 | -----
mongo-express_1 |
mongo-express_1 | Mongo Express server listening at http://0.0.0.0:8081
mongo-express_1 | Server is open to allow connections from anyone (0.0.0.0)
mongo-express_1 | basicAuth credentials are "admin:pass", it is recommended
you change this in your config.js!
mongo_1          | 2019-08-12T22:54:08.221+0000 I NETWORK [listener]
connection accepted from 172.19.0.3:40942 #2 (1 connection now open)
mongo_1          | 2019-08-12T22:54:08.247+0000 I NETWORK [conn2] received
client metadata from 172.19.0.3:40942 conn2: { driver: { name: "nodejs",
version: "2.2.24" }, os: { type: "Linux", name: "linux", architecture: "x64",
version: "4.9.184-linuxkit" }, platform: "Node.js v8.16.0, LE, mongodb-core:
2.1.8" }
mongo-express_1 | Database connected
mongo_1          | 2019-08-12T22:54:08.384+0000 I ACCESS [conn2]
Successfully authenticated as principal root on admin from client
172.19.0.3:40942
mongo-express_1 | Admin Database connected
mongo_1          | 2019-08-12T22:54:08.398+0000 I NETWORK [listener]
```

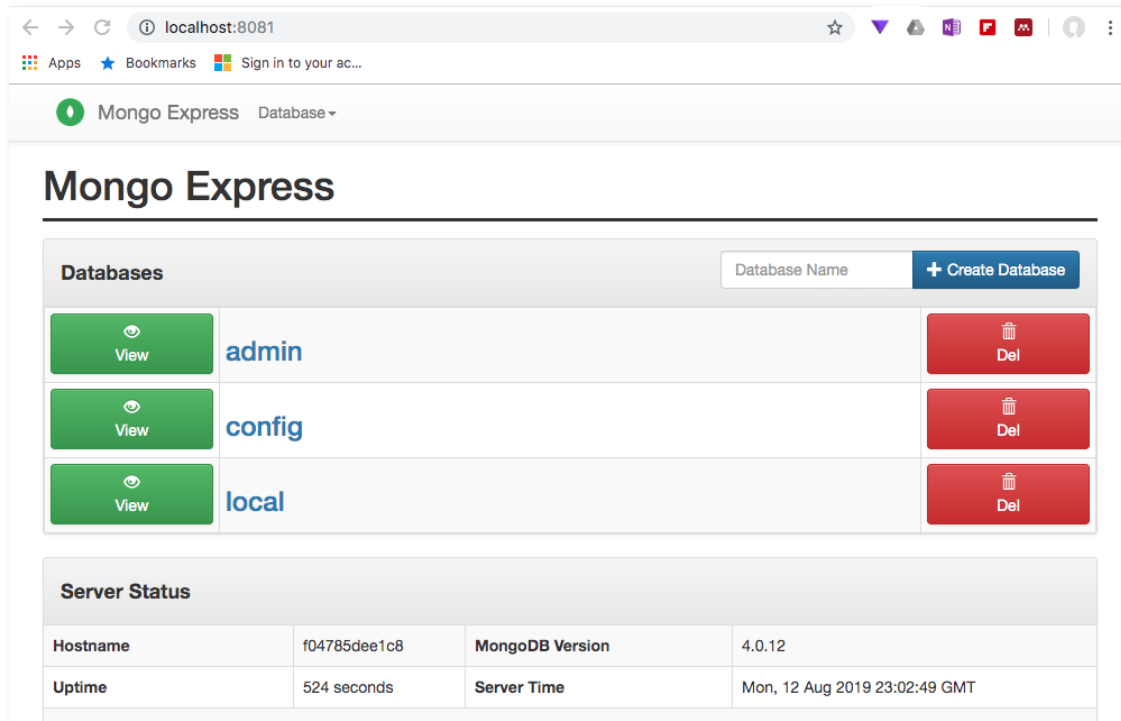
Week 8

```
connection accepted from 172.19.0.3:40944 #3 (2 connections now open)
mongo_1          | 2019-08-12T22:54:08.505+0000 I ACCESS    [conn3]
Successfully authenticated as principal root on admin from client
172.19.0.3:40944
mongo_1          | 2019-08-12T22:59:33.041+0000 I NETWORK    [listener]
connection accepted from 172.19.0.3:40950 #4 (3 connections now open)
mongo_1          | 2019-08-12T22:59:33.141+0000 I ACCESS    [conn4]
Successfully authenticated as principal root on admin from client
172.19.0.3:40950
mongo-express_1  | GET / 200 168.802 ms - 8954
mongo-express_1  | GET /public/css/bootstrap-theme.min.css 200 7.987 ms -
23409
mongo-express_1  | GET /public/css/bootstrap.min.css 200 9.814 ms - 121200
mongo-express_1  | GET /public/css/style.css 200 12.850 ms - 1883
mongo-express_1  | GET /public/index-512f467a07c538127931.min.js 200 13.133
ms - 1042
mongo-express_1  | GET /public/img/mongo-express-logo.png 200 15.140 ms -
17847
mongo-express_1  | GET /public/vendor-47ad38d08af7dbfa26be.min.js 200 12.806
ms - 126314
mongo-express_1  | GET /public/img/gears.gif 200 3.903 ms - 50281
mongo-express_1  | GET /public/fonts/glyphicons-halflings-regular.woff2 200
1.370 ms - 18028
```

Browse to

<http://localhost:8081>

Week 8



mongo express menu

To exercise some of the docker-compose commands; in a separate terminal window still pointing to the mongo1 directory:

```
docker-compose -f stack.yml ps
```

Terminal output shows the running containers

Name	Command	State	Ports

mongo1_mongo-express_1	tini -- /docker->8081/tcp	Up	0.0.0.0:8081-
mongo1_mongo_1	entrypoint ... docker-entrypoint.sh mongod	Up	27017/tcp

```
docker-compose -f stack.yml pause
```

Terminal output shows pausing:

```
Pausing mongo1_mongo-express_1 ... done
Pausing mongo1_mongo_1          ... done
```

```
docker-compose -f stack.yml ps
```

Confirms the paused status

Name	Command	State	Ports

mongo1_mongo-express_1	tini -- /docker->8081/tc	Paused	0.0.0.0:8081-
mongo1_mongo_1	entrypoint ... docker-entrypoint.sh mongod	Paused	p 27017/tcp

```
docker-compose -f stack.yml unpause
```

Unpauses the containers

```
Unpausing mongo1_mongo_1      ... done
Unpausing mongo1_mongo-express_1 ... done
```

Stop containers (so you could switch off machine and take a break).

```
docker-compose -f stack.yml stop
```

```
Stopping mongo1_mongo_1      ... done
Stopping mongo1_mongo-express_1 ... done
```

Now try restarting the pc.

```
docker-compose -f stack.yml start
```

```
Starting mongo      ... done
Starting mongo-express ... done
```

```
http://localhost:8081
```

All back up ok.

Don't use 'docker-compose -f stack.yml down' till you are actually through using the container.

Create and populate a database

In this section I am following the tutorial example at [Mozilla developer](#)

An app will be developed to administer a book library.

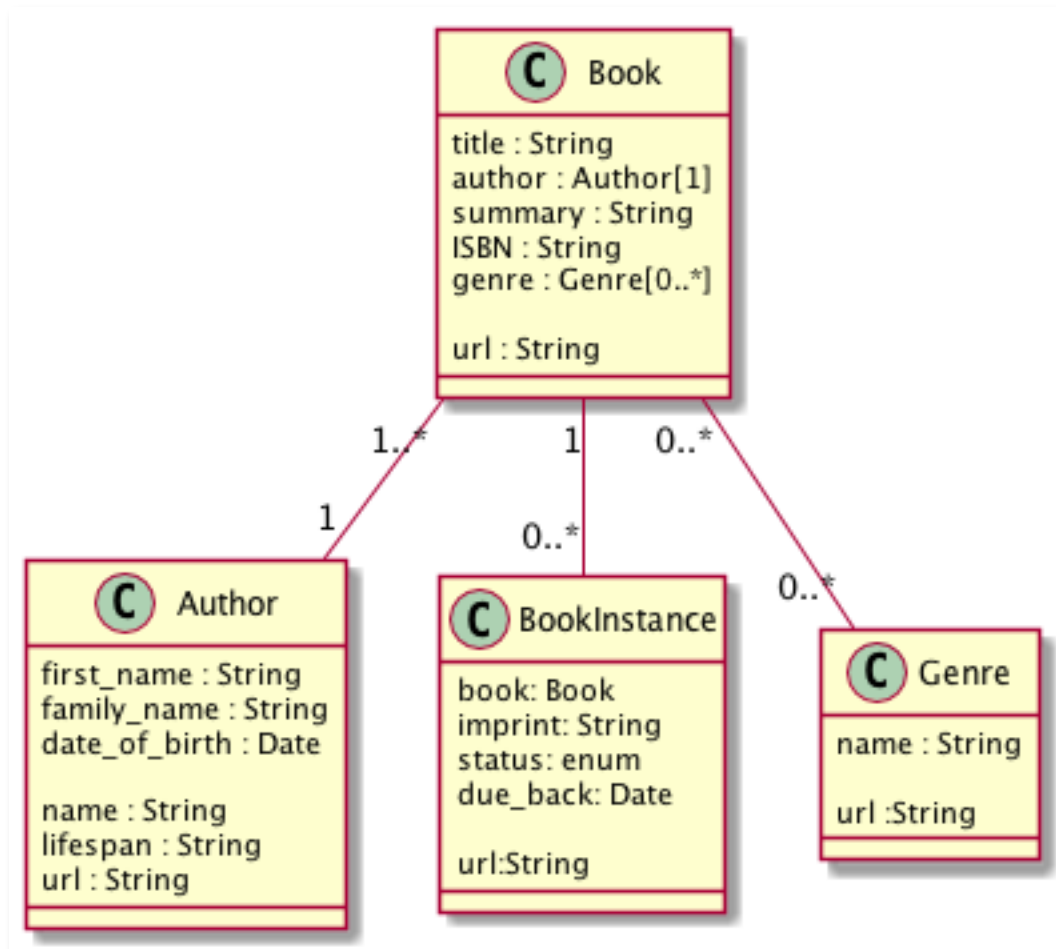
For this purpose the objects which need to be described in the database are:

1. Book: described by title author ISBN etc
2. BookInstance: The library can hold several copies of one book. When it is loaned each book instance will have a due date for return.
3. Author: name and birth/death dates
4. Genre: Crime, Sci-fi, Romance etc

The relationships between these are:

- 1 book may have 0..many instances
- 1..many books may be written by 1 author (ignore multiple authors for the time being)
- 0..many books may match 0..many genres (a book might fit more than one genre or defy categorisation)

This relationship is captured in the UML diagram:



database uml

So examples of initial database entries would be:

Book:

- title: 'The Name of the Wind (The Kingkiller Chronicle, #1)',
- summary: 'I have stolen princesses back from sleeping barrow kings. I burned down the town of Trebon. I have spent the night with Felurian and left with both my sanity and my life. I was expelled from the University at a younger age than most people are allowed in. I tread paths by moonlight that others fear to speak of during day. I have talked to Gods, loved women, and written songs that make the minstrels weep.',
- ISBN: '9781473211896',

Week 8

- author: authors[0],
- genre: [genres[0],]

BookInstance

- book: books[0],
- imprint: 'London Gollancz, 2014.',
- due_back: false,
- status: 'Available'

Author:

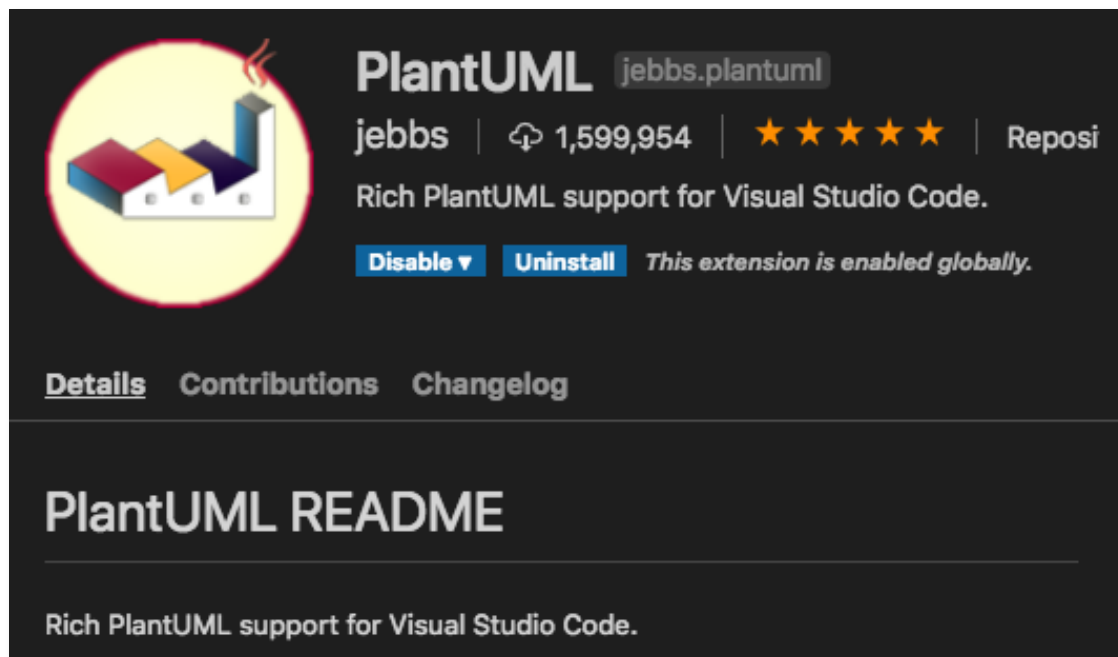
- first_name: 'Patrick',
- family_name: 'Rothfuss',
- d_birth: '1973-06-06',
- d_death: false

Genre:

name: "Fantasy"

Creating a UML diagram

As a brief sidenote, visual studio code is able to create a UML diagram through the addition of a plug in [plantUML](#).



database uml

A tutorial at code project [UML Made Easy with PlantUML & VS Code](#) is a further useful guide to using the plug in.

Week 8

Working in a new folder named `uml_diagrams` the following file For the diagram of the library database is saved as `library.plantuml`.

```
@startuml demo
```

```
class Book {
    title : String
    author : Author[1]
    summary : String
    ISBN : String
    genre : Genre[0..*]

    url : String
}
```

```
class Author {
    first_name : String
    family_name : String
    date_of_birth : Date

    name : String
    lifespan : String
    url : String
}
```

```
class BookInstance {
    book: Book
    imprint: String
    status: enum
    due_back: Date

    url:String
}
```

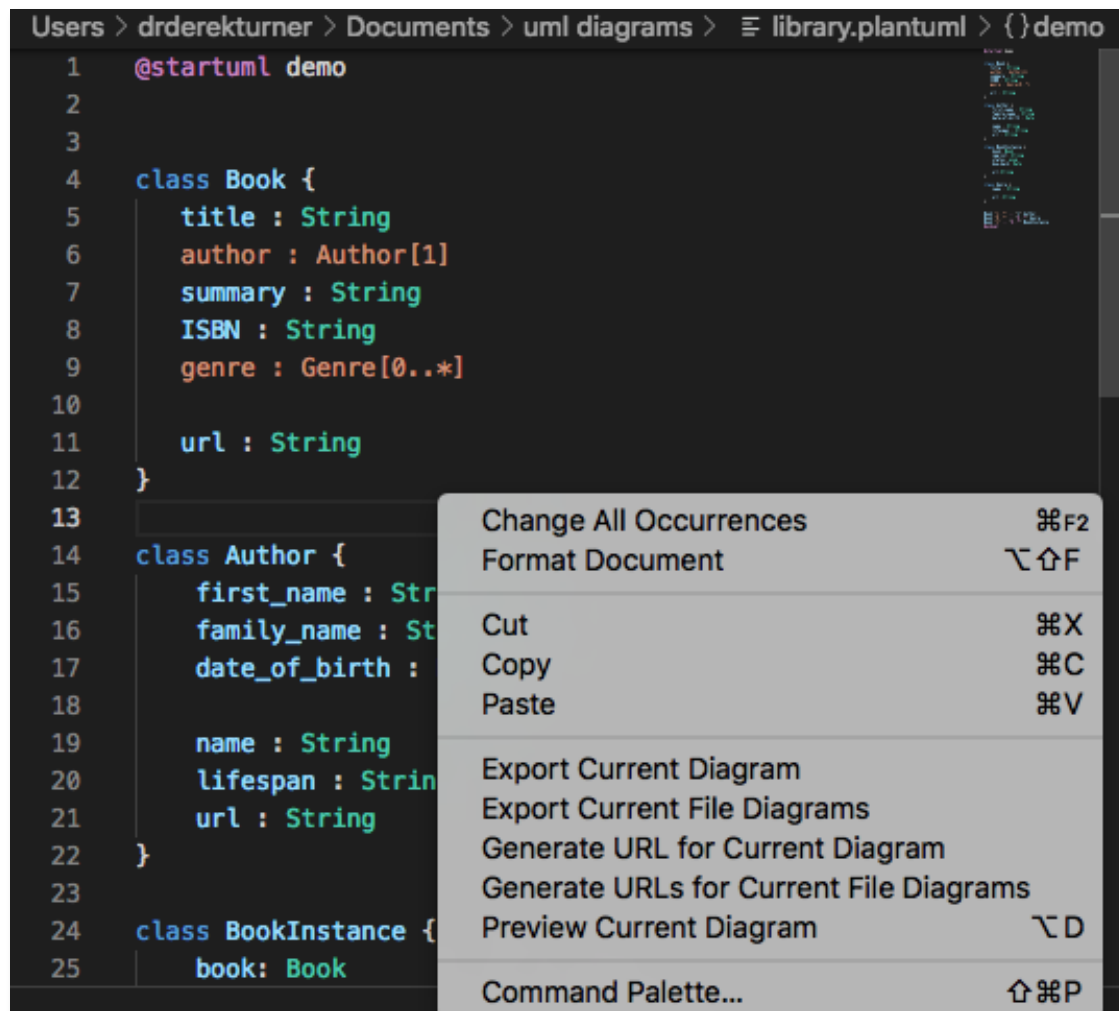
```
class Genre {
    name : String

    url :String
}
```

```
Book "1..*" -- "1" Author
Book "0..*" -- "0..*" Genre
Book "1" -- "0..*" BookInstance
@enduml
```

Right clicking over the file opens a menu to see a preview of the diagram, which will change as the file is edited. Then to print the digram to the desktop in your chosen format. (The

plug in works well if you have Java on your machine, otherwise you may need additional software).

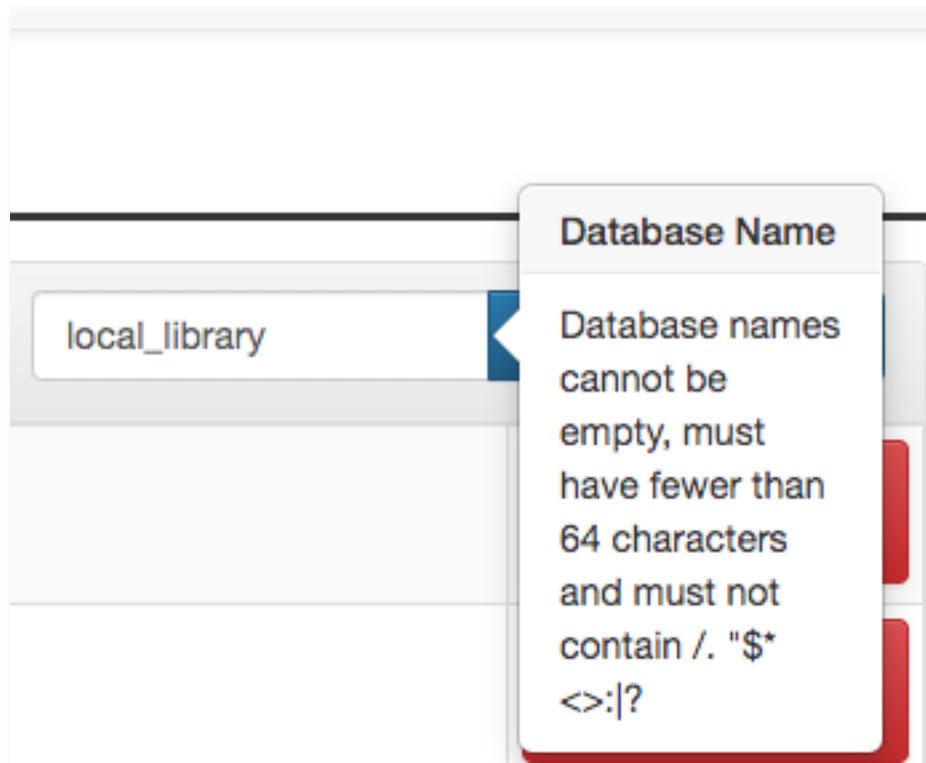


database uml

Back to the database

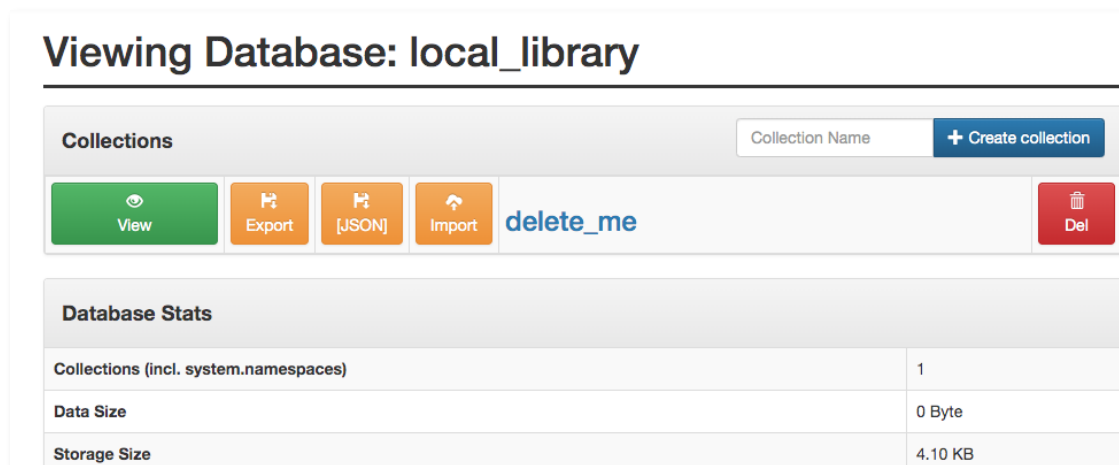
<http://localhost:8081>

The mongo-express web page interface allows the user to create view and edit databases. View the Mongo-Express page and click on create Database



create database

View the database and click on import.



view database

Providing a file now which has a JSON formatted single database object will load it to the library.

Working item by item would take ages to add the initial data to seed the database, mongo-express is only useful for correcting and maintaining the database once it is uploaded.

Docker provides a method to populate the database which is described in the environmental variables section of the docker hub [mongo page](#).

Ensure that the mongo container is down.

```
docker-compose -f stack.yml down
```

Terminal output confirms

```
Stopping mongo1_mongo_1          ... done
Stopping mongo1_mongo-express_1  ... done
Removing mongo1_mongo_1          ... done
Removing mongo1_mongo-express_1  ... done
Removing network mongo1_default
```

Now add a lines to the environment section of the docker compose file stack.yml

```
environment:
  MONGO_INITDB_ROOT_USERNAME: root
  MONGO_INITDB_ROOT_PASSWORD: example
  MONGO_INITDB_DATABASE: local_library
volumes:
  - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
```

The environment variable MONGO_INITDB_DATABASE contains the name of the database which is required to be initialise. If this is omitted the default database will be 'db'.

When the docker compose file comes 'up' the volumes line will cause the file mongo-init to be copied into a reserved name mount point '/docker-entrypoint-initdb.d/'.

When the container is brought up, *if the database is empty*, the scripts in the '/docker-entrypoint-initdb.d/' mountpoint will be run. Only one script is used in this example, but it is possible to use multiple scripts. These scripts can initialise the database.

Starting and stopping the app again does not cause the database to be altered.

The mongo image itself creates a volume so that the database persists on the host machine.

Add a file mongo-init.js to the mongo1 folder which will initialise the data for the local-library database. Working through this file:

First set up variables, the database will have four collections as previously described, the data which is going into each of these will be stored to an array.

```
let error = false;

let genres = [];
let authors = [];
let books = [];
let bookinstances = [];
```

Each container will need a function to allow data passed in to be added as a JSON object to the matching array. When the function is called it will also push the JSON data into the res array. No need to declare the res array, it is a keyword representing the response which the script will send to the database. The overall strategy is to store command objects to the res array (in the correct order) and then output the response.

The function for the genre collection is simplest. The genre just has one name element.

```
function genreCreate(name) {
  genredetail = { name: name };
  genres.push(genredetail);
  res.push(db.genre.insert(genredetail));
}
```

The res array is defined.

The first lines are commands to create the four empty connections. Note that although the database is called local_library, the reference in this file is still to the default database name 'db'.

```
let res = [
  db.book.drop(),
  db.author.drop(),
  db.bookinstance.drop(),
  db.genre.drop(),
```

The next group of commands create the indexes for each of the collections. Note that you need the index name and a number, which in this example is always 1. If a field is to contain unique data this should be noted.

```
db.book.createIndex({ title: 1 }, { unique: true }),
db.book.createIndex({ summary: 1 }),
db.book.createIndex({ author: 1 }),
db.book.createIndex({ isbn: 1 }),
db.book.createIndex({ genre: 1 }),
```

The definition of res is now closed off

```
]
```

The functions to create the data can now be called. For authors all the data is known, so the calls are straightforward.

```
authorCreate("Patrick1", "Rothfuss", "1973-06-06", false);
```

For other collections, such as 'book' the related data needs to be included.

```
bookCreate("The Wise Man's Fear ... the unassuming pub landlord.",
'9788401352836', authors[0], [genres[0],]);
```

Note that this refers to authors[0], so the authorCreate() calls must all be done before the bookCreate() calls. In an SQL database a reference authorID would be inserted here.

However in a noSQL database every document must be complete and stand alone, so the full JSON object description is added at this point. If you request a book, you don't need to make another request to find out who the author is with authorID 'xxxx'.

Because a book could match zero to many genres an array of the matching genre objects is passed.

A noSQL database does not work with structured query language, but it does have a structure and there are relationships between data elements.

Last stage is to hope for no errors and send the response to the database.

```
printjson(res);
```

```
if (error) {  
  print("Error, exiting");  
  quit(1);  
}
```

The full listing for mongo-init.js for the local_library tutorial is:

```
let error = false;  
  
let genres = [];  
let authors = [];  
let books = [];  
let bookinstances = [];  
  
function authorCreate(first_name, family_name, d_birth, d_death) {  
  authordetail = {  
    first_name: first_name,  
    family_name: family_name,  
    d_birth: null,  
    d_death: null  
  };  
  if (d_birth != false) authordetail.d_birth = d_birth;  
  if (d_death != false) authordetail.d_death = d_death;  
  authors.push(authordetail);  
  res.push(db.author.insert(authordetail));  
}  
  
function genreCreate(name) {  
  genredetail = { name: name };  
  genres.push(genredetail);  
  res.push(db.genre.insert(genredetail));  
}  
  
function bookCreate(title, summary, isbn, author, genre) {  
  bookdetail = {  
    title: title,
```

Week 8

```
        summary: summary,
        author: author,
        isbn: isbn,
        genre: null
    };
    if (genre !== false) bookdetail.genre = genre;
    books.push(bookdetail);
    res.push(db.book.insert(bookdetail));
}

function bookInstanceCreate(book, imprint, due_back, status) {
    bookinstancedetail = {
        book: book,
        imprint: imprint,
        due_back: null,
        status: null
    };
    if (due_back !== false) bookinstancedetail.due_back = due_back;
    if (status !== false) bookinstancedetail.status = status;
    bookinstances.push(bookinstancedetail);
    res.push(db.bookinstance.insert(bookinstancedetail));
}

let res = [
    db.book.drop(),
    db.author.drop(),
    db.bookinstance.drop(),
    db.genre.drop(),

    db.book.createIndex({ title: 1 }, { unique: true }),
    db.book.createIndex({ summary: 1 }),
    db.book.createIndex({ author: 1 }),
    db.book.createIndex({ isbn: 1 }),
    db.book.createIndex({ genre: 1 }),

    db.author.createIndex({ first_name: 1 }),
    db.author.createIndex({ family_name: 1 }),
    db.author.createIndex({ d_birth: 1 }),
    db.author.createIndex({ d_death: 1 }),

    db.bookinstance.createIndex({ book: 1 }),
    db.bookinstance.createIndex({ imprint: 1 }),
    db.bookinstance.createIndex({ due_back: 1 }),
    db.bookinstance.createIndex({ status: 1 }),

    db.genre.createIndex({ name: 1 })
];

authorCreate("Patrick", "Rothfuss", "1973-06-06", false);
```

Week 8

```
authorCreate("Ben", "Bova", "1932-11-8", false);
authorCreate("Isaac", "Asimov", "1920-01-02", "1992-04-06");
authorCreate("Bob", "Billings", false, false);
authorCreate("Jim", "Jones", "1971-12-16", false);

genreCreate("Fantasy");
genreCreate("Science Fiction");
genreCreate("French Poetry");

bookCreate(
    "The Name of the Wind (The Kingkiller Chronicle, #1)",
    "I have stolen princesses back from sleeping barrow kings. I burned down the town of Trebon. I have spent the night with Felurian and left with both my sanity and my life. I was expelled from the University at a younger age than most people are allowed in. I tread paths by moonlight that others fear to speak of during day. I have talked to Gods, loved women, and written songs that make the minstrels weep.",
    "9781473211896",
    authors[0],
    [genres[0]]
);
bookCreate(
    "The Wise Man's Fear (The Kingkiller Chronicle, #2)",
    "Picking up the tale of Kvothe Kingkiller once again, we follow him into exile, into political intrigue, courtship, adventure, love and magic... and further along the path that has turned Kvothe, the mightiest magician of his age, a legend in his own time, into Kote, the unassuming pub landlord.",
    "9788401352836",
    authors[0],
    [genres[0]]
);
bookCreate(
    "The Slow Regard of Silent Things (Kingkiller Chronicle)",
    "Deep below the University, there is a dark place. Few people know of it: a broken web of ancient passageways and abandoned rooms. A young woman lives there, tucked among the sprawling tunnels of the Underthing, snug in the heart of this forgotten place.",
    "9780756411336",
    authors[0],
    [genres[0]]
);
bookCreate(
    "Apes and Angels",
    "Humankind headed out to the stars not for conquest, nor exploration, nor even for curiosity. Humans went to the stars in a desperate crusade to save intelligent life wherever they found it. A wave of death is spreading through the Milky Way galaxy, an expanding sphere of lethal gamma ...",
    "9780765379528",
    authors[1],
    [genres[1]]
);
```

Week 8

```
);
bookCreate(
    "Death Wave",
    "In Ben Bova's previous novel New Earth, Jordan Kell led the first human
mission beyond the solar system. They discovered the ruins of an ancient
alien civilization. But one alien AI survived, and it revealed to Jordan Kell
that an explosion in the black hole at the heart of the Milky Way galaxy has
created a wave of deadly radiation, expanding out from the core toward Earth.
Unless the human race acts to save itself, all life on Earth will be wiped
out...",
    "9780765379504",
    authors[1],
    [genres[1]]
);
bookCreate("Test Book 1", "Summary of test book 1", "ISBN111111", authors[4],
[
    genres[0],
    genres[1]
]);
bookCreate(
    "Test Book 2",
    "Summary of test book 2",
    "ISBN222222",
    authors[4],
    false
);

bookInstanceCreate(books[0], "London Gollancz, 2014.", false, "Available");
bookInstanceCreate(books[1], " Gollancz, 2011.", "2020-06-06", "Loaned");
bookInstanceCreate(books[2], " Gollancz, 2015.", false, false);
bookInstanceCreate(
    books[3],
    "New York Tom Doherty Associates, 2016.",
    false,
    "Available"
);
bookInstanceCreate(
    books[3],
    "New York Tom Doherty Associates, 2016.",
    false,
    "Available"
);
bookInstanceCreate(
    books[3],
    "New York Tom Doherty Associates, 2016.",
    false,
    "Available"
);
bookInstanceCreate(
    books[4],
```

Week 8

```
    "New York, NY Tom Doherty Associates, LLC, 2015.",
    false,
    "Available"
);
bookInstanceCreate(
    books[4],
    "New York, NY Tom Doherty Associates, LLC, 2015.",
    false,
    "Maintenance"
);
bookInstanceCreate(
    books[4],
    "New York, NY Tom Doherty Associates, LLC, 2015.",
    false,
    "Loaned"
);
bookInstanceCreate(books[0], "Imprint XXX2", false, false);
bookInstanceCreate(books[1], "Imprint XXX3", false, false);

printjson(res);

if (error) {
    print("Error, exiting");
    quit(1);
}
```

When writing an initialisation script, use the editor to spot syntax errors.

Then when the docker compose file is brought up, watch the detail of the terminal output to see error messages such as.

```
mongo_1          | 2019-08-16T09:31:01.728+0000 E QUERY    [js]
ReferenceError: book is not defined :
mongo_1          | @/docker-entrypoint-initdb.d/mongo-init.js:89:19
mongo_1          | failed to load: /docker-entrypoint-initdb.d/mongo-init.js
```

It is useful to have one terminal window for docker compose up and one for docker compose down. This saves multiple chang directory commands.

Eventually you should have success with

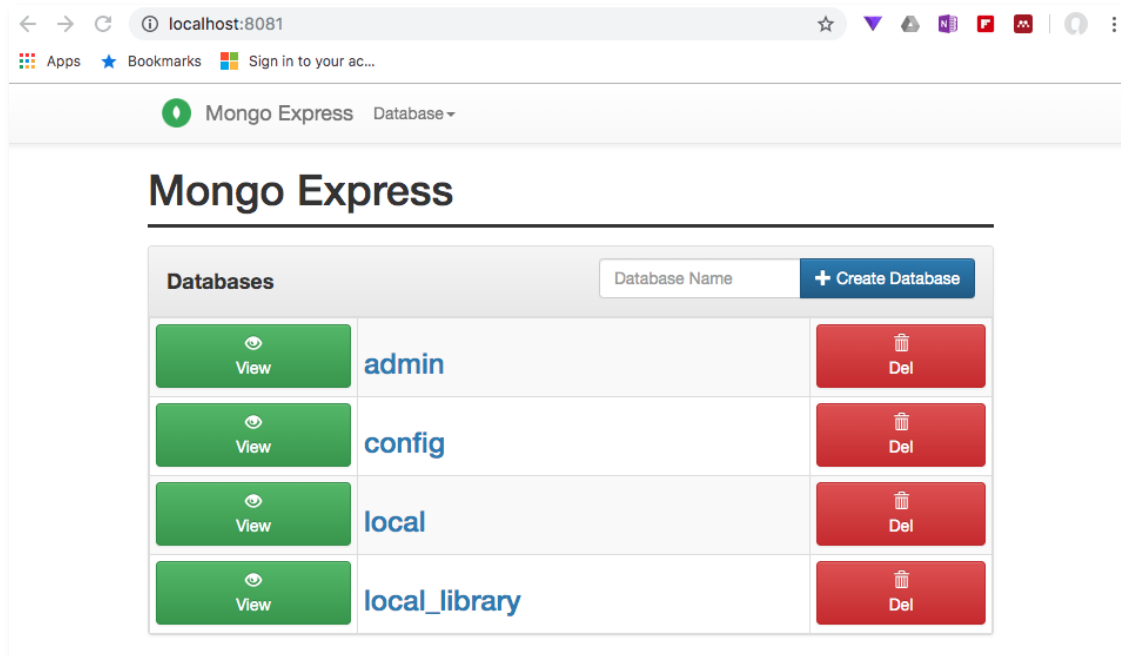
```
docker-compose -f stack.yml up
```

Inspect the database at

```
http://localhost:8081/
```

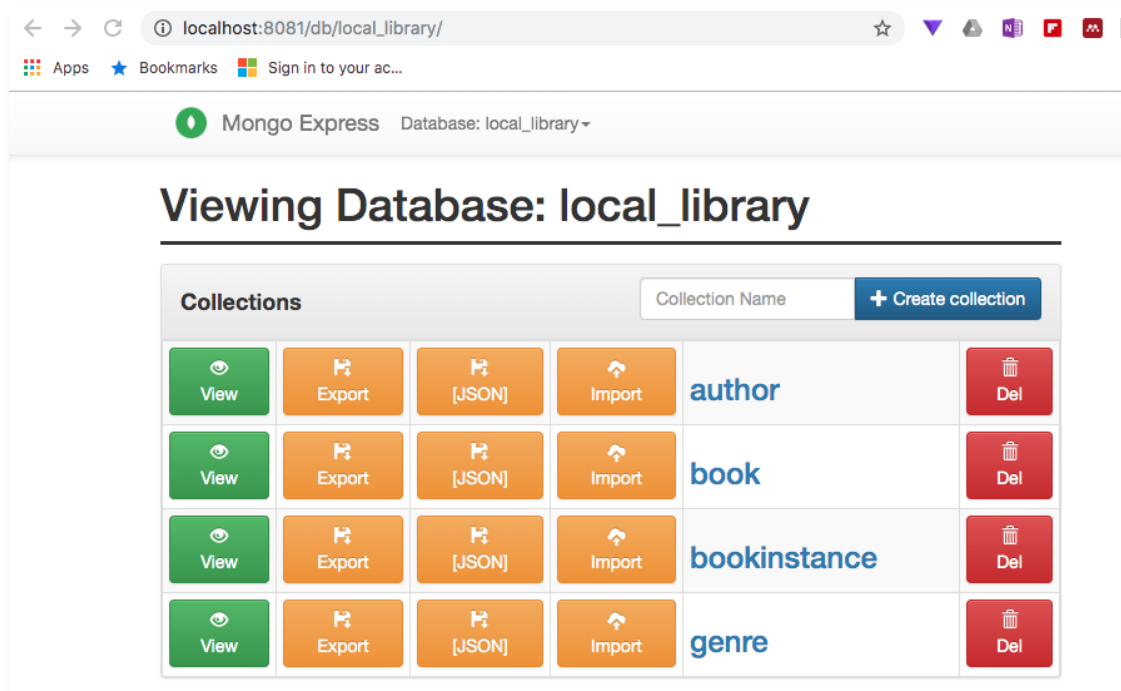
Note that the local_library database has been created:

Week 8



librarycreated

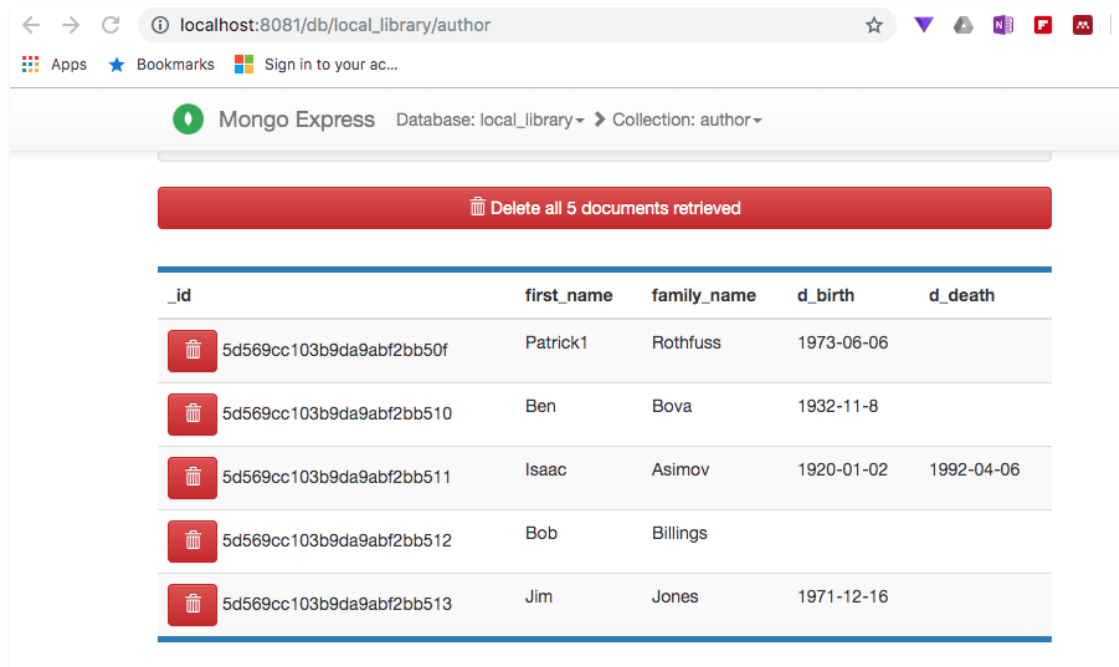
View the library to see the collections.



collectionscreated

View the details of the authors collection

Week 8

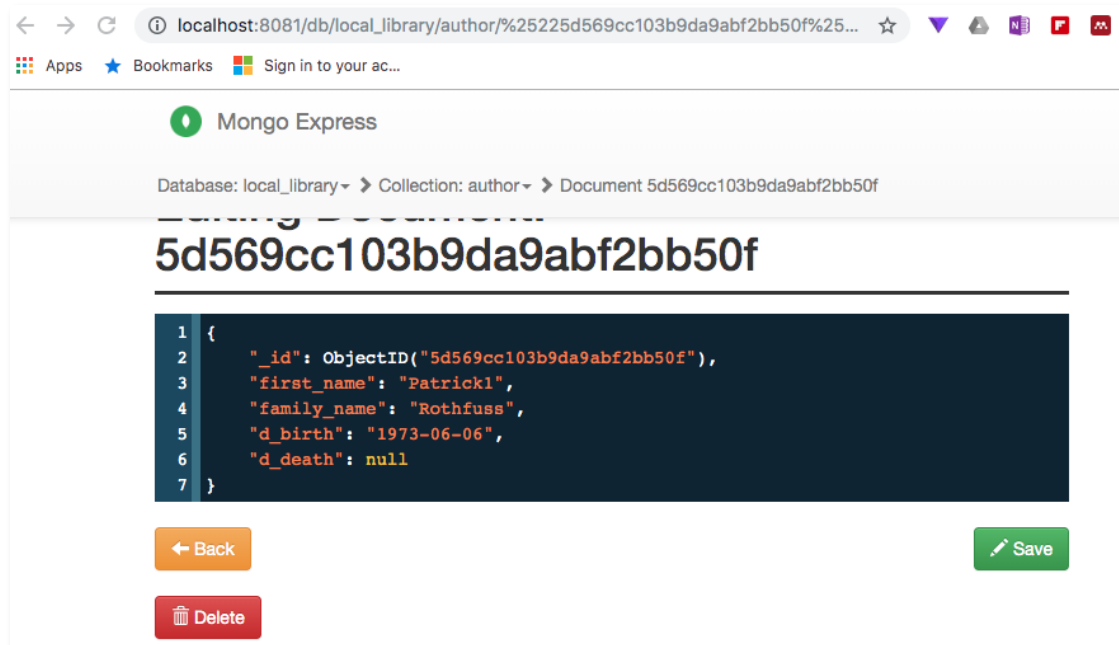


The screenshot shows the Mongo Express interface for the 'local_library' database and 'author' collection. A red banner at the top indicates 'Delete all 5 documents retrieved'. Below is a table with 5 rows of author data. Each row has a trash icon in the first column, followed by the _id, first_name, family_name, d_birth, and d_death.

	_id	first_name	family_name	d_birth	d_death
	5d569cc103b9da9abf2bb50f	Patrick1	Rothfuss	1973-06-06	
	5d569cc103b9da9abf2bb510	Ben	Bova	1932-11-8	
	5d569cc103b9da9abf2bb511	Isaac	Asimov	1920-01-02	1992-04-06
	5d569cc103b9da9abf2bb512	Bob	Billings		
	5d569cc103b9da9abf2bb513	Jim	Jones	1971-12-16	

authors populated

Double click an individual entry line to see the JSON object representation including a unique id.



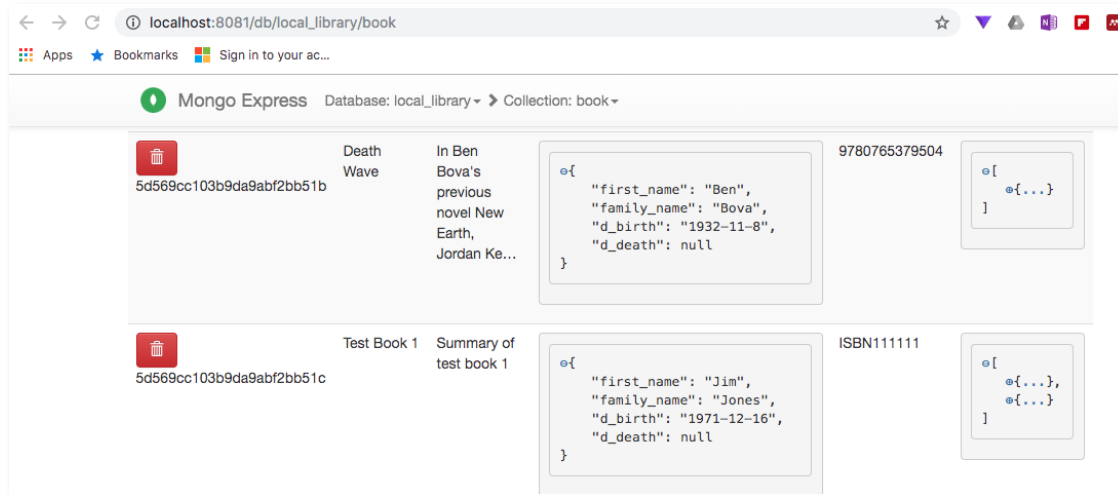
The screenshot shows the Mongo Express interface for the 'local_library' database and 'author' collection, displaying the JSON representation of a document with the unique id '5d569cc103b9da9abf2bb50f'. The document is shown in a dark-themed code editor. Below the code editor are buttons for 'Back', 'Save', and 'Delete'.

```
1 {
2   "_id": ObjectID("5d569cc103b9da9abf2bb50f"),
3   "first_name": "Patrick1",
4   "family_name": "Rothfuss",
5   "d_birth": "1973-06-06",
6   "d_death": null
7 }
```

json author

See that the documents in the book collection contain objects representing the full author details for each book.

Week 8

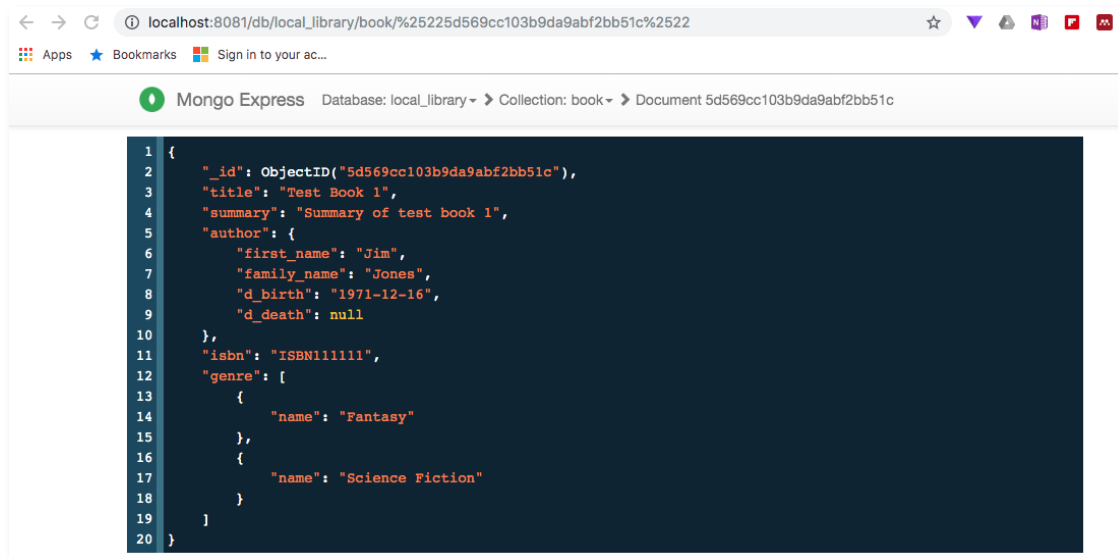


book details

Note that the author field contains an object. Each object has a clickable icon. A small circle with a horizontal line. Click this to see the json view of that object.

Note that 'test book 1' had two genres attributed so you can see both represented as {}. Each inner object has a small circle icon with a cross to open the JSON view of that object.

Double click the 'test book 1' line to see how this is represented in JSON



book with genres

Spend some time inspecting the data and check out the import export functions of mongo express.

Use docker compose stop/start to switch the application on and off. Using docker compose down/up will remove everything and restore back to this seeded database.

Week 8

exercise

Design and create your own database for a simple example.

references

[How To Build a Node.js Application with Docker](#) sharks webpage

[express](#)

[How to create a REST API with Express.js in Node.js](#)

[Creating a skeleton website](#)