Lab 8
MongoDB via Mongoose

CSCI2720 Building Web Applications

Dr. Chuck-jee Chau
chuckjee@cse.cuhk.edu.hk

Agenda

- MongoDB account
- Connecting via Mongoose
- Schema and Model
- Handling GET/POST for database access
- Optional materials:
  - Setting up MongoDB
  - Accessing MongoDB

MongoDB account

- For the labs and assignments, we have prepared a MongoDB account on the cloud
- You can use this account without worrying about MongoDB server setup
- Yet, if you prefer, you can also try MongoDB Atlas, and let us know if you run into issues
  - More guidance at the end of the lab slides

MongoDB account

- Please check your @link email for ***CSCI2720/ESTR2106 MongoDB account info***
- You can find these inside:
  - Database
  - DB Username
  - DB Password
  - MongoDB connect string
- These are all you need in your Node.js apps to connect

Connecting via Mongoose

- You can use the same Node.js setup you tried in Lab 7
- Make a copy of your lab 7 work, and edit the js file
- Install Mongoose support

```
npm install mongoose
```

- In your js file, add this

```
var mongoose = require('mongoose');
mongoose.connect('server URL');
```

- The ***server URL*** is the "connect string"
  - `mongodb+srv://user:pass@server/userdb`
  - Use your *username*, *password* and *userdb* there
- You may also use the ***options*** parameter as in the lecture slides

Connecting via Mongoose

- Then you can show appropriate messages depending on whether the connection is successful
- Try and restructure your Lab 7 js file like this:

```
const express = require('express');
const app = express();
const mongoose = require('mongoose');
mongoose.connect('your actual server URL');

const db = mongoose.connection;
// Upon connection failure
db.on('error', console.error.bind(console,
'Connection error:'));
// Upon opening the database successfully
db.once('open', function () {
  console.log("Connection is open...");
  /* ... Lab 7 work on app.get() and app.post() */
  app.get('/event/:eventId/loc/:locId', (req, res)
=> { ... });
  ...
})
const server = app.listen(3000);
```

- Run/reload your server and see if the database connection is successful

Schema and Model

- Our aim is to store some *event information* into the database
- Insert the following schema definition after the db connection is made successfully

```
const EventSchema = mongoose.Schema({
    eventId: { type: Number, required: true, unique: true },
    name: { type: String, required: true },
    loc: { type: String },
    quota: { type: Number }
});
```

- And create a model based on this schema

```
const Event = mongoose.model('Event', EventSchema);
```

Handling GET requests

- To avoid confusion, you may comment out the GET request handler in Lab 7, since we will make a simpler one here first

```
app.get('/event/:eventId', (req,res) => {
  Event.findOne(
    {eventId: req.params['eventId']},
    'eventId name loc quota',
    (err, e) => {
      if (err)
        res.send(err);
      else
      res.send("This is event "+e.eventId+":<br>\n" +
       "Event name: " + e.name + "<br>\n" +
       "Event location: " + e.loc + "<br>\n" +
       "Event quota: " + e.quota + "<br>\n" +
       "Ref: " + e);
  });
});
```

- ***Check your syntax very carefully!!***
- You would see an error when visiting `/event/123` in the browser, since there is NOTHING in our database!

Adding test data

- For testing purpose, we can add some sample data when a special URL is called
  - For example, at `/testevent`

```
app.get('/testevent', (req,res) => {
  Event.create({
    eventId: 123,
    name: 'A nice event',
    loc: 'Not in CUHK',
    quota: 2
  }, (err, e) => {
  if (err) res.send(err);
  else res.send(e);
  });
});
```

- Then, run the Node app and access `/testevent` in your browser
- If successful, you'll see the event data on screen, and now you can try `/event/123`

{"eventId":123,"name":"A nice event","loc":"Not in CUHK","quota":2,"_id":

Handling POST requests

- Next, we will set up a form for new events
- You can comment out the POST request handler you made in Lab 7, and use this instead

```
app.post('/event', (req,res) => {
  Event.create({
    eventId: req.body['eventId'],
    name: req.body['name'],
    loc: req.body['loc'],
    quota: req.body['quota']
  }, (err,e) => {
     if (err)
       res.send(err);
     else
       res.send("Ref: " + e);
  });
});
```

- You can get this form for submitting the POST request

  https://www.cse.cuhk.edu.hk/~chuckjee/2720lab8/form.html

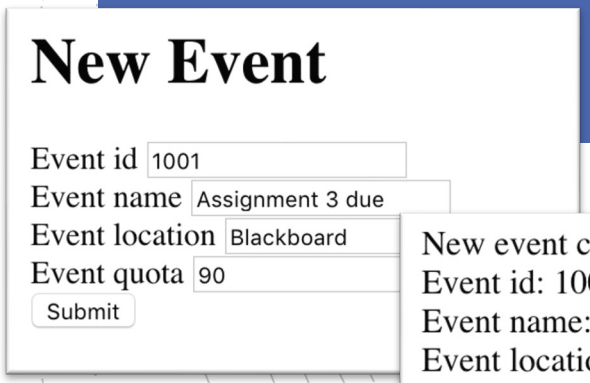
  - It is better to save the file and run on your local computer!

Handling POST requests

- For this POST handler to work, you need the ***body parser***

```
const bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({extended:false}));
```

- Check that you have the correct URL for the ***form action*** in the HTML file
- Are you able to improve the result after posting, into this form below?



New event created:
Event id: 1001
Event name: Assignment 3 due
Event location: Blackboard
Event quota: 90
Ref: { _id: 5bd9c8b83e6bf83ea97c9c09, eventId: 1001, name: 'Assignment 3 due', loc: 'Blackboard', quota: 90, __v: 0 }

Quick Summary

- A number of techniques are involved in this lab:
  - Node.js + Express
  - MongoDB + Mongoose
  - HTML form, HTTP GET/POST
- It is very easy to get lost!
- It is likely you get confused by syntax
- ***Thank you for your patience!***

Submission

- No submission is needed for labs
- What you have done could be useful for your further exploration or the upcoming assignment
- **Please keep your own code safely**

Setting up MongoDB

- It is possible to set up a MongoDB server on your computer
  - Follow the instructions here for your OS: https://docs.mongodb.com/manual/administration/install-community/
- At the end of installation, you shall set up MongoDB to be run as a *service*

Setting up MongoDB

- Another possible thing you can do is to try the Atlas Cloud
  - Follow the instructions here: *https://docs.atlas.mongodb.com/tutorial/deploy-free-tier-cluster/*
  - An account is needed on Atlas
- To connect to the cloud db, you can add `0.0.0.0` to the IP access list, which mean ANYWHERE
  - That's not so secure…

Accessing MongoDB

- Besides using Node.js, there are two ways to directly manipulate MongoDB database
  - Compass (GUI)
    - https://docs.mongodb.com/compass
  - MongoDB Shell (CLI)
    - https://docs.mongodb.com/mongodb-shell
- There are lots more to learn.

  Good luck!